C:\Users\Manfred Stueber\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Logo_Hauptcampus_Informatik.eps

Darstellung der Topologie eines industriellen Kommunikationsnetzwerks

Master-Abschlussarbeit

Betreuer: Prof. Dr. Georg Rock und Frank Fährmann

Hattersheim, 26.4.2018

Vorwort und Danksagung

Die vorliegende Masterarbeit ist im Rahmen meiner Tätigkeit als Junior Softwareentwickler in der Firma Hilscher GmbH entstanden. Ich war in der Abteilung „User Interface Group“ beschäftigt. Die Arbeit wurde in Zusammenarbeit mit der Entwicklung der Konfiguration Software „CommunicationStudio“ entwickelt, da das entstehende Software als Plugin [[1]](#footnote-1)verwendet werden soll.

Trotz versuchter Fokussierung auf die wesentlichen Aspekte überschreitet diese Arbeit

die ursprünglich geplante Seitenanzahl. Ein wesentlicher Grund dafür ist, dass ich dem

Grundsatz der Transparenz des Vorgehens, wie er mir in meinem Fernstudium vermittelt

wurde, so gut wie möglich gerecht werden wollte. Sehr viele Schritte meiner

theoretischen und praktischen Arbeit habe ich daher versucht, zur intersubjektiven

Nachvollziehbarkeit für den Leser ausführlich und verständlich zu dokumentieren und

hoffe, dass mir dies gelungen ist.

Zunächst möchte ich mich an dieser Stelle bei all denjenigen bedanken, die mich während der Masterarbeit unterstützt und motiviert haben.

Ganz besonders gilt dieser Dank Herrn Frank Fährmann, der mich und meine Arbeit betreut hat. Nicht nur gaben Sie mir immer wieder durch kritisches Hinterfragen wertvolle Hinweise – auch Ihre kontinuierliche Motivation hat einen großen Teil zur Vollendung dieser Arbeit beigetragen. Sie haben mich dazu gebracht, über meine Grenzen hinaus zu denken. Vielen Dank für die Geduld und Mühen.

Darüber hinaus gebührt jedem mein Dank, der durch Korrekturen, Verbesserungsvorschläge und hilfreiche Diskussionen zur Qualität dieser Arbeit beigetragen hat. Gleiches gilt für meine neugeborene Tochter und meine Frau, die mir in dieser Zeit eine unersetzbare seelische und moralische Stütze waren.

Auch möchte ich bei allen Kollegen, die ich im Rahmen meines Projektstudiums interviewt habe, danken. Ohne ihre Kooperation hätte ich meine Arbeit nicht realisieren können.

Kurzfassung

Bei der Konfiguration von Systeme in der industrielleren Kommunikation unterschiedlichen Geräten und Protokollen werden zur Qualitätssicherung und die Produktivität geeignete Modelle benötigt. Der Schwerpunkt liegt hierbei auf die Visualisierung bzw. Überwachung der Systeme und der damit verbundenen Topologien immer wieder abzufahren. Für diese Zwecke hat die Firma Hilscher auf Basis eines Desktops bzw. Webanwendung einen Konfiguration Software mit dem Namen „CommunicationStudio“ entwickelt. Die Nutzung dieser Software ist jedoch umständlich in der Bedienung und muss ein Anwender erst in dem Produktionsbereich gehen um die Anordnung bzw. die Topologie des Systems zu beobachten.

Im Rahmen dieser Arbeit soll für den Anwender eine Web-Komponente in HTML5 erstellt werden. Diese dient als Plug-In für CommunicationStudio.

Die grafische Oberfläche soll es hierbei ermöglichen, die einzelnen Funktionen der Visualisierung abstrakt und intuitiv zu nutzen. Des Weiteren sollen einzelnen Darstellung-Szenarien abgespeichert und geladen werden können, die unabhängig vom Protokoll die Topologie industrieller Kommunikationsnetzwerkes ermitteln und darstellen. Eine Automatisierung Anordnung der Geräte und Verbindungen soll ebenfalls zur Verfügung stehen. Besonders Augenmerk sollten hierbei auch eine einfache zukünftige Erweiterbarkeit der Software mit neuen Funktionen gelegt werden.

Im ersten Schritt der Arbeit soll anhand einer Analyse der Protokolle PROFIBUS und PROFINET *IRT* [[2]](#footnote-2)mögliche Formen der Topologie ermittelt werden. Auf dieser Basis soll ein Konzept zur Ermittlung und Darstellung entwickelt werden. Abschließend wird das Konzept in die Praxis umgesetzt

Inhaltsverzeichnis

[1 Einleitung 9](#_Toc512521639)

[1.1 Grund 9](#_Toc512521640)

[1.2 Ziele 10](#_Toc512521641)

[1.3 Aufgaben 10](#_Toc512521642)

[1.4 Firmenvorstellung 12](#_Toc512521643)

[2 Grundlagen 13](#_Toc512521644)

[2.1 Grundlegende Begrifflichkeiten 13](#_Toc512521645)

[2.2 Grundlegende Netzwerktopologien 14](#_Toc512521646)

[2.2.1 Kennwerte einer Netzwerktopologie 15](#_Toc512521647)

[2.2.2 Linien-Topologie 16](#_Toc512521648)

[2.2.3 Ring-Topologie 17](#_Toc512521649)

[2.2.4 Stern-Topologie 17](#_Toc512521650)

[2.2.5 Baum-Topologie 18](#_Toc512521651)

[2.2.6 Vermaschtes Netz 19](#_Toc512521652)

[2.3 Netzwerke-Protokolle 20](#_Toc512521653)

[2.3.1 PROFIBUS 20](#_Toc512521654)

[2.3.2 PROFINET- IRT 21](#_Toc512521655)

[3 Die Ausgangssituation der bestehenden Softwares 22](#_Toc512521656)

[3.1 Was ist „Communication Studio“? 22](#_Toc512521657)

[3.2 Befragungsmethoden 24](#_Toc512521658)

[3.3 Anlass der Plug-In „*Web-Topology Editor*“ 24](#_Toc512521659)

[4 Anforderungsanalyse 25](#_Toc512521660)

[4.1 Funktionale Anforderungen 25](#_Toc512521661)

[4.1.1 Muss*-*Kriterien 25](#_Toc512521662)

[4.1.2 Kann-Kriterien 26](#_Toc512521663)

[4.2 Nichtfunktionale Anforderungen 27](#_Toc512521664)

[4.2.1 Muss-Kriterien 27](#_Toc512521665)

[4.2.2 Kann-Kriterien 27](#_Toc512521666)

[5 Evaluation der Technologien 28](#_Toc512521667)

[5.1 Clientseitige relevante Frameworks 28](#_Toc512521668)

[5.1.1 Angular 5 29](#_Toc512521669)

[5.1.2 Vue.js 30](#_Toc512521670)

[5.1.3 React [8] 30](#_Toc512521671)

[5.2 Vergleichskriterien 30](#_Toc512521672)

[5.2.1 Lernkurve und Kosten 30](#_Toc512521673)

[5.2.2 Kommunikationsmechanismen 30](#_Toc512521674)

[5.2.3 Programmierkonzept 31](#_Toc512521675)

[5.3 GoJS 31](#_Toc512521676)

[5.3.1 Einführung 31](#_Toc512521677)

[5.3.2 Vorteile und Nachteile von GoJS 32](#_Toc512521678)

[5.4 Bootstrap 33](#_Toc512521679)

[5.5 ASP.NET CORE 34](#_Toc512521680)

[5.5.1 REST 34](#_Toc512521681)

[5.5.2 ASP.NET Core 2.0 35](#_Toc512521682)

[*6* Grundprinzipien von *WebTopologyEditor* 36](#_Toc512521683)

[6.1 FDT 36](#_Toc512521684)

[6.2 TopologyEditor grundlegende Konzepte 36](#_Toc512521685)

[6.3 Kernszenarien 37](#_Toc512521686)

[6.3.1 Offline Szenario 38](#_Toc512521687)

[6.3.2 Online Szenario 38](#_Toc512521688)

[6.4 Datendarstellung (Prozessdaten, Bedienungsdaten) 38](#_Toc512521689)

[6.5 Zukunftsaspekte (evtl. Erweiterungen) 40](#_Toc512521690)

[7 Entwurf 41](#_Toc512521691)

[7.1 Architektur der *TopologyEditor* 41](#_Toc512521692)

[7.2 Datenmodell 42](#_Toc512521693)

[7.3 DtmApi Wrapper 43](#_Toc512521694)

[7.4 Ansicht der TopologyEditor 43](#_Toc512521695)

[8 Realisierung 45](#_Toc512521696)

[8.1 Vorgehensweise 45](#_Toc512521697)

[8.2 Verwendete Werkzeugen 45](#_Toc512521698)

[8.3 Ausgewählte Implementierungsaspekte 45](#_Toc512521699)

[8.3.1 Serverseitige Kodierung 46](#_Toc512521700)

[8.3.2 Clientseitige Kodierung 47](#_Toc512521701)

[8.3.3 Darstellung der Topologie anzeigen 47](#_Toc512521702)

[8.3.4 Darstellung der Topologie aktualisieren 47](#_Toc512521703)

[8.3.5 Darstellung der Topologie speichern und laden 47](#_Toc512521704)

[8.3.6 Fehlerbehandlung einer Topologie 47](#_Toc512521705)

[9 Systemtest 48](#_Toc512521706)

[9.1 Anwendungsfall: Topologie anzeigen 48](#_Toc512521707)

[9.2 Anwendungsfall: Gerät identifizieren 49](#_Toc512521708)

[9.3 Anwendungsfall: Gerät Parameter lesen 49](#_Toc512521709)

[9.4 Anwendungsfall: Geräte manuell verbinden 49](#_Toc512521710)

[9.5 Anwendungsfall: Verbindung Parameter lesen 49](#_Toc512521711)

[9.6 Anwendungsfall: Topologie exportieren 49](#_Toc512521712)

[9.7 Anwendungsfall: Topologie importieren 49](#_Toc512521713)

[10 Zusammenfassung und Ausblick 50](#_Toc512521714)

[10.1 Zusammenfassung 50](#_Toc512521715)

[10.2 Ausblick 50](#_Toc512521716)

[Literatur 51](#_Toc512521717)

[Index 52](#_Toc512521718)

[Glossar 53](#_Toc512521719)

[Erklärung der Kandidatin / des Kandidaten 54](#_Toc512521720)

Abbildungsverzeichnis

[Abbildung 1: Haupteingang Hilscher Gesellschaft für Systemautomation mbH 11](#_Toc511225559)

[Abbildung 2: Linien-Topologie 15](#_Toc511225560)

[Abbildung 3: Ring-Topologie 16](#_Toc511225561)

[Abbildung 4: Stern-Topologie 17](#_Toc511225562)

[Abbildung 5: Baum-Topologie 18](#_Toc511225563)

[Abbildung 6: das vermaschte Netz zwischen den einzelnen Netzgerät 18](#_Toc511225564)

[Abbildung 7: Beispielhafte Topologie zur Verdeutlichung der PROFIBUS-System in der industriellen Kommunikation mit einem Master-Slave-Verfahren 20](#_Toc511225565)

[Abbildung 8: Übersichtschema *CStudio* mit zukünftiger *Web-Topology Editor* 22](#_Toc511225566)

[Abbildung 10: Die Output JSON-Dateien für die Mini-Topologie 30](#_Toc511225567)

[Abbildung 11: Typische zusammengesetzte Anwendungsarchitektur mit dem Angular Framework 33](#_Toc511225568)

[Abbildung 12: Die wichtigsten Eigenschaften des Diagramms und ihre Interaktion 34](#_Toc511225569)

[Abbildung 13: Diagramm zeigt den Grundlegenden Entwurf der TopologyEditor 38](#_Toc511225570)

[Abbildung 14: Domänenmodell der Elemente des TopologyEditor 39](#_Toc511225571)

[Abbildung 15: Klassendiagramm eines Beispiel für Fassaden mit feldbusabhängigen Protokollen 40](#_Toc511225572)

Tabellenverzeichnis

[Tabelle 1: Termin- und Meilensteinplanung für die Projektarbeit **Fehler! Textmarke nicht definiert.**](#_Toc510696010)

# Einleitung

Unter Kommunikation versteht man den Austausch von Daten zweier Einheiten. Diese Einheiten können zwei Softwaremodule sein, die sich auf einem Gerät oder auch auf unterschiedlichen Geräten befinden. In einem Netzwerk können sich mehrere Module oder Geräte befinden, die Daten miteinander austauschen. Neben der logischen Darstellung des Datenflusses gibt aus auch eine physikalische Darstellung der Geräte ohne Berücksichtigung des Datenflusses. In dieser Arbeit soll nur die physikalische Darstellung, d. h. die Verknüpfung der Geräte eines industriellen Netzwerks betrachtet werden. Die logischen Verbindungen sollen in Zukunft nur eine weitere Sicht auf das Netzwerk sein. In der industriellen Kommunikation kommen nun noch verschiedene Anforderung hinzu wie z.B. Echtzeitverhalten und Übertragungssicherheit. Grundsätzlich unterstützt die Firma *Hilscher* alle gängigen Netzwerkprotokolle. In den vergangenen Jahren ist eine Vielzahl neuer und Protokolle hinzugekommen. Die Anforderungen an Netzwerke, die aus der Industrie kommen, steigen stetig. Durch die permanent steigenden Anforderungen steigt auch der Aufwand in der Konfiguration, der neu entstehenden Netzwerkeprotokolle. Durch die geringe Anzahl an Netzwerkprotokollen, konnte man früher davon ausgehen, dass sich der Inbetriebnehmer eines Netzwerkes mit der Konfiguration auskannte. Heute muss man davon ausgehen, dass der Anwender kein detailliertes Wissen über die Konfiguration des Netzwerkes besitzt. Aus diesem Grund ist es umso wichtiger die Konfiguration für den Anwender einfach und intuitiv zu gestalten. Ein weiterer aber nicht unwichtiger Grund ist die Reduzierung des Support-Aufkommens, der sich aus einer selbsterklärenden Software ergibt. Zur Konfiguration eines Netzwerkes gehören noch weitere feldbusprotokollabhängige Parameter und Einstellungen, die hier an dieser Stelle nicht betrachtet werden und als gegeben vorausgesetzt werden wie z.B. die Übertragungsgeschwindigkeit der Daten. Die grafischen Darstellungen sind ein ideales Mittel für die Visualisierung von komplexeren Topologien. Durch die Visualisierung der Topologie des Netzwerkes kann dem Anwender ein schnellerer und leichterer Überblick sowie Einstieg in das Netzwerk gegeben werden. Mit dieser Arbeit sollen die Grundlagen für einen vereinheitlichten Topologie-Editor geschaffen werden, damit nicht für jedes entstehende industrielle Netzwerkprotokoll ein neuer Topologie-Editor programmiert werden muss. Ziel dabei ist, in einem Topologie-Editor verschiedene Netzwerke darzustellen. Für die Firma Hilscher sind, neben dem Vorteil der Kostenreduzierung und der Wiederverwendbarkeit der Software, das einheitliche „*Look and Feel*“ sowie das gleiche Bedienkonzept der Software für den Kunden weitere Vorteile.

## Grund

Der Grund für diese Arbeit ist, das die Hilscher Gesellschaft für Systemautomation eine Möglichkeit sucht, komplexe industrielle Netzwerke grafisch darzustellen und zu konfigurieren. Die Ausgangssituation ist, dass es momentan keine Möglichkeit gibt, eine physikalische Topologie eines Netzwerkes unabhängig vom Netzwerkprotokoll zu erstellen. Die entstehende Komponente soll später als Plug-In für die bereits bestehende Software Communication Studio der Firma Hilscher integriert werden. Wie in der Einleitung erwähnt, gibt es verschiedene Gründe eine unabhängige Topologie-Komponente zu erstellen. Neben der Wiederverwendbarkeit und auch der Kostenreduktion ist meiner Meinung nach einer der wichtigsten Gründe diese Komponente als grafische Oberfläche zu entwerfen. Der Grund ist, dass der Mensch diese Art der visuellen Informationen am schnellsten Erfassen und Verstehen kann. Da ich auch selbst großes Interesse grafische Oberflächen zu entwerfen und zu entwickeln und darüber hinaus faszinieren mich zunehmend Entwicklung von Software, habe ich mich für diese Masterarbeit entschieden.

## Ziele

Eines der Ziele, die mit dieser Masterarbeit erreicht werden sollen, ist, dass die Kunde der Firma Hilscher einfach und mit wenig Aufwand seine Aufgaben, das Netzwerk zu konfigurieren, erfüllen kann. Hierzu soll sich der Anwender nicht erst das entsprechende Fachwissen aneignen müssen. Wie auch schon in der Einleitung erwähnt, sollte es das Ziel sein, das Support-Aufkommen zu reduzieren. Andere Ziele wie Wiederverwendbarkeit oder Integration in eine bestehende Software spielen natürlich auch eine wichtige Rolle.

Im bisherigen Entwicklungsfokus der Firma Hilscher standen zunächst nur die Kernfunktion zur Konfiguration eines industriellen Netzwerkes der neuen Software „Communication Studio“ im Vordergrund. Durch die gezielte Integration eines Software Plug-In-Moduls „Topology Editor“ in die Software „Communication Studio“ können positive auf Modellierung eines Netzwerkes einwirken. Dabei stellt sich die Frage, wie das Plug-In für die bestehende Software entwickelt werden soll, insbesondere unter der Voraussetzung der oben genannten Ziele sowie auch geringer struktureller Veränderungen der bestehenden Software.

Meine persönlichen Ziele, die ich mit dieser Masterarbeit erreichen möchte, sollen ein tieferes und umfassendes Wissen über die Netzwerktopologie in der industrielleren Kommunikation sein. Weiterhin soll mir die Implementierung dieser Software die Aufgaben und Pflichten der Rolle eines Senior Softwareentwickler in einem Unternehmen aufzeigen. Im Rahmen dieser Arbeit möchte ich mir fundierte Fachkenntnisse aneignen und so die Fähigkeit erlangen, komplexe Vorhaben zu planen und umzusetzen.

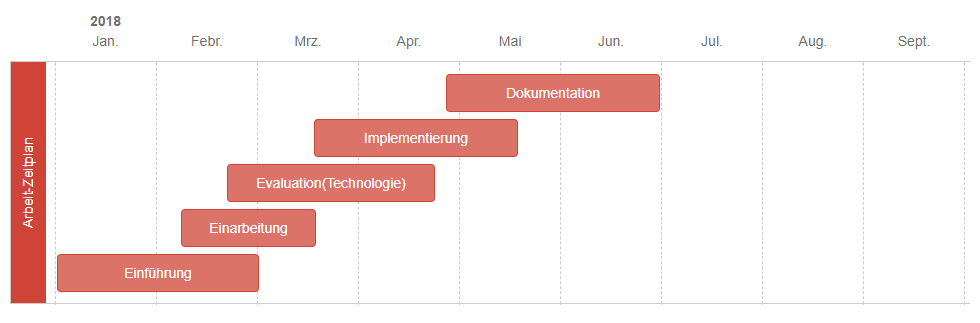
Aus diesem Grund soll im Rahmen der vorliegenden Masterarbeit eine Möglichkeit zur Integration des Plug-In in die Konfigurations-Software „Communication Studio“ entstehen.

## Aufgaben

Durch in Abschnitt 1.2 vordefinierte Ziele ergeben sich nun folgende Aufgaben im Rahmen dieser Masterarbeit. Zum einen sollen verschiedenen Software-Komponenten entstehen, die unabhängig vom Netzwerkprotokoll die Topologie des industriellen Kommunikationsnetzwerkes darstellen und zum anderen auch die Bearbeitung in einem gewissen Rahmen zulassen.

Da dieses Themengebiet sehr komplex und umfangreich ist und über viel Fachwissen voraussetzt, erfolgt im ersten Schritt eine Einführung und Einarbeitung in das Thema der Arbeit, bevor anhand einer Analyse der Protokolle PROFIBUS und PROFINET IRT Formen der Topologie und verschiedene Darstellungsarten analysiert werden. Auf Basis der durch die Evaluierung gewonnenen Erkenntnisse wird im zweiten Schritt ein Konzept zur Darstellung entwickelt. Nach der Erstellung des Konzeptes erfolgt dann die Implementierung. Die Implementierung erfolgt dann in mehreren Iterationsstufen.

Da die Arbeit nur über einen bestimmten zeitlichen Rahmen verfügt und in dieser Zeit prototypenhaft entwickelt werden soll, habe ich mir im Vorfeld Gedanken über eine zeitliche Einteilung gemacht, um nicht die Übersicht und das Ziel der Fertigstellung des Projektes zu verlieren. Ich habe die Masterarbeit in verschieden Phasen eingeteilt und mit einem Start und Ende versehen.



[Abbildung 1:](#Eins_Abbildung) Arbeit Zeitplan für die Entwicklung von Plug-In „*Topology-Editor“*

Abbildung 1 zeigt Beginn und Ende der einzelnen festgelegten Projektphasen. Die Festlegung erfolgt aufgrund des festgelegten Fertigstellungstermin. Neben den Projektterminen wurden zu diesem Zeitpunkt auch die wichtigen Meilensteine festgelegt. Tabelle 1 zeigt beides: die Termin- und Meilensteinplanung.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Beschreibung** | **Ressource** | **Beginn** | **Ende** | **Meilenstein** |
| Einführung | Erklärungen und Erläuterungen der Aufgaben | Projektleiter gemeinsam mit allen Beteiligten | 01.01.2018 | 01.03.2018 | Ziele der Arbeit verstehen |
| Einarbeitung | Erarbeitung der Grundlagen und Theorie für den Fachbereich | Student (ich) | 15.01.2018 | 14.02.2018 | Topologien kennenlernen,  Protokolle einarbeiten |
| Evaluation | Erstellung einer Konzeption | Student mit Gruppenleiter | 12.02.2018 | 15.04.2018 | Konzept erstellen |
| Realisierung | Erstellung des Prototyp | Student  Entwicklungsumgebung VS 2017 | 25.03.2018 | 25.05.2017 | Prototype erstellen |
| Dokumentation | Erstellen von Text und Grafiken | Student Word, Drucker | 02.05.2018 | 15.06.2018 | Schriftlichen Teil Masterarbeit erstellen |

Tabelle : Termin- und Meilensteinplanung für die Projektarbeit

## Firmenvorstellung

Die Firma Hilscher Gesellschaft für Systemautomation mbH besteht seit 1986 und beschäftigt heute über 260 Mitarbeiter an 10 Standorten weltweit. Mit der Philosophie des kontinuierlichen Wachstums aus eigener Kraft, ist das Unternehmen ein verlässlicher Partner für seine Kunden. Alle Prozesse sind nach ISO 9001 und 14001 zertifiziert.

Die Kernkompetenz von der Firma Hilscher ist die Technologie, Entwicklung und Produktion von industriellen Kommunikationslösungen für die moderne Fabrikautomation. Die Produkte reichen von PC-Karten und Gateways über OEM-Aufsteckmodule bis hin zu leistungsfähigen ASICs mit den dazugehörigen Protokoll-Stacks. Diese werden weltweit zur Kommunikation zwischen Automatisierungsgeräten und Steuerungen eingesetzt. Im Bereich PC-Karten ist die Firma Marktführer. Ein derart umfassendes Lösungs-Portfolio für Feldbusse und Real-Time-Ethernet ist das Alleinstellungsmerkmal der Firma Hilscher. Hilscher bietet auch ASICs basierte die netX-Technologie für Gerätehersteller an, inklusive Entwicklungs-Dienstleistungen und kundenspezifischer Baugruppen-Fertigung. In diesem Bereich ist die Firma nicht nur anerkannter Systempartner der großen Hersteller, sondern zählt auch eine Vielzahl von Ingenieurbüros und Systemintegratoren zu seinen Kunden. Außerdem ist Hilscher in allen Feldbus- und Real-Time-Ethernet-Organisationen vertreten



Abbildung : Haupteingang Hilscher Gesellschaft für Systemautomation mbH

# Grundlagen

In diesem Kapitel werden für das Verständnis der Arbeit wichtige Grundlagen eingeführt. Zunächst wird eine allgemeine Definition der Fachbegriffe in der industrielleren Kommunikation vorgenommen. Des Weiteren folgt eine Darstellung der Anordnungsarten an Netzwerke und eine kurze Einführung in die Netzwerk-Protokolle bei Automation Besonders die PROFINET und Profibus-Protokoll. Daran schließt sich eine Übersicht über die Funktionsweise von PROFINET und Profibus-Technik und eine Diskussion der Gründe für einen Einsatz dieser Technologie für Feldbusse in der Industrie. Dieser Abschnitt endet mit einer Darstellung verschiedener Ansätze, um Echtzeitfähigkeit mit PROFINET zu erreichen.

## Grundlegende Begrifflichkeiten

Bevor wir über das Thema starten, erklären wir meisten genutzte Terminologien in der Industrieller Kommunikation.

* **Netzwerk**. Als *Netzwerk* bezeichnet man den Verbund mehrere Rechner oder Rechnergruppen zum Zweck der Datenkommunikation. Sie können nach der räumlichen Ausdehnung (Größe) bzw. durch die Art der Leitungsführung oder nach Übertragungsgeschwindigkeit unterschieden werden.
* **Ethernet–Technik:** Unter *Ethernet* versteht man eine Datenübertragungstechnik, die es ermöglicht Daten innerhalb geschlossener Netzwerke zwischen verschiedenen Geräte zu Übertragen. Es umfasst die Festlegung für spezielle genormte Kabel und Steckverbindungen sowie für Paketformat und Protokolle. Ethernet ist in der Norme IEE802.3 standarisiert worden, und ist zur weitverbreitetsten LAN-Technologie geworden. Ethernet bildet die Basis für Übertragungsprotokolle wie z.B. TCP IP, PROFINET oder EtherCat um nur die bekanntesten zu nennen.
* **Netzwerkprotokoll:** Damit Kommunikationspartner in einem Netzwerk überhaupt miteinander kommunizieren können, müssen sie bestimmte Vereinbarungen und Regeln einhalten. Diese Regeln sind die *Protokolle*. Ein einzelnes Protokoll arbeitet immer einen bestimmten Funktionsbereich ab, der bei der Datenkommunikation der Funktionalität einer Schicht des OSI-Referenzmodell entsprechen kann.
* **OSI-Referenzmodell:** Das *OSI-Referenzmodell* [1] der Internationalen Standardisierungs-Organisation (ISO) wurde ab 1977 als Grundlage für die Bildung von Kommunikationsstandards entworfen. Ziel von der Open Systems Interconnection (OSI) ist die Kommunikation in heterogenen Netzen.
* **Feldbus**: Ein *Feldbus* verbindet in einer Anlage verschiedene Sensoren (Messfühler) und Aktoren (Stellglieder) zwecks Kommunikation mit einem übergeordneten Steuerungsgerät (SPS). Dabei senden und empfangen in der Regel mehrere Teilnehmer nahezu gleichzeitig über ein und dieselbe Leitung. Damit es zu keinen Datenkollisionen kommt, muss also festgelegt werden wer (Kennung) was (Messwert, Befehl) wann (Initiative) sagt.
* **Knoten & Kanten**: Knoten repräsentieren die Kommunikationspartner und die Kante die Verbindung in der Graph Theorie
* **Kennwerte**: Sind die Eigenschaften einer Topologie
* **Topologie**: Teilgebiet der Mathematik, welches ursprünglich diejenigen Eigenschaften geometrischer Gebilde behandelt, die bei umkehrbar eindeutigen, stetigen und totalen Abbildungen erhalten bleiben, das sogenannte Homomorphismus [2] . In der industrielleren Kommunikation beschreibt man die Topologie die Verbindungen der teilnehmenden Geräte untereinander, die den Datenverkehr ermöglichen.
* **Controller**: Ein Controller ist die übergeordnete Steuerung, die typischerweise das Prozessabbild und das Anwenderprogramm enthält. Er ist der aktive Teil bei der der Kommunikation und parametriert und konfiguriert die angeschlossenen Devices. Er führt den zyklischen/azyklischen Datenaustausch, sowie die Alarmbearbeitung mit den prozessnahen Devices durch. [3]
* **Supervisor**: Dies kann ein Programmiergerät, Personal Computer oder Human Man Interface-Gerät zu Inbetriebsetzungs- oder Diagnosezwecken sein. Um beispielsweise den Prozess für Testwecke zu steuern oder die Diagnoseauswertung zu übernehmen, kann ein Supervisor temporär die Funktionsweise eines Controllers übernehmen.
* **Device**: Ein Device ist der passive Station bei der Kommunikation, der gemäß Protokoll die Prozessdaten an die übergeordnete Steuerung überträgt und kritische Anlagenzustände beispielerweise Diagnose und Alarme meldet.

## Grundlegende Netzwerktopologien

Die Topologie eines Netzwerkes ist entscheidend für seine Ausfallsicherheit: Nur, wenn alternative Wege zwischen den Knoten existieren, bleibt bei Ausfällen einzelner Verbindungen die Funktionsfähigkeit erhalten. Es gibt dann neben dem Arbeitsweg einen oder mehrere Ersatzwege (oder auch Umleitungen).

Die richtige Wahl der Topologie und deren Pflege kann die Performance [[3]](#footnote-3)des Netzes beeinflussen, dies Bedeutetet, die Kenntnis der Topologie eines Netzes ist außerdem nützlich zur Bewertung seiner Performance, sowie der Investitionen und für die Auswahl geeigneter Hardware.

Es wird zwischen physikalischer und logischer Topologie unterschieden. Die physikalische Topologie beschreibt den Aufbau der Netzverkabelung; die logische Topologie den Datenfluss zwischen den Endgeräten.

Topologien werden grafisch (nach der Graphentheorie) mit Knoten und Kanten dargestellt.

In großen Netzen findet man oftmals eine Struktur, die sich aus mehreren verschiedenen Topologien zusammensetzt.

Wir werden die logische Topologie nicht eingehen da kein Teil der Arbeit ist. Dieser Abschnitt endet mit der Liste der bekanntesten physikalischen Topologien in der Industrie.

### Kennwerte einer Netzwerktopologie

**Durchmesser**

Der Durchmesser einer Topologie beschreibt die maximale direkte Entfernung zwischen zwei Knoten in Hops[[4]](#footnote-4). Damit ist er ein direktes Maß für die zu erwartenden maximalen Transferzeiten, d.h. je größer der Durchmesser, desto größer die Transferzeit im ungünstigsten Fall.

**Grad**

Der Grad einer Topologie gibt die Anzahl der Links pro Knoten an. Diese kann für jeden Knoten gleich oder verschieden sein. Haben alle Knoten einer Topologie den gleichen Grad, so ist die Topologie regulär, was sich vorteilhaft auf das Netzwerk auswirkt. Außerdem beschreibt der Grad indirekt, welche Kosten man zum Aufbau der Topologie aufbringen muss. Je höher der Grad, desto höher die Kosten.

**Bisektionsweite**

Die Bisektionsweite gibt die minimale Anzahl von Links an, die durchschritten werden müssen, um ein Netz mit N Knoten in zwei Netze mit jeweils N/2 Knoten zu teilen. Damit ist sie ein Maß für die Leistungsfähigkeit eines Netzes, da in vielen Algorithmen die Knoten der einen Netzhälfte mit den Knoten der anderen Hälfte kommunizieren. Je niedriger also die Bisektionsweite, desto ungünstiger wirkt sich dies auf den Zeitbedarf für den Datenaustausch zwischen beiden Netzhälften aus.

**Symmetrie**

Bei einer symmetrischen Topologie sieht das Netz von jedem Betrachtungspunkt (Knoten/Links) gleich aus, d.h. es existieren für Knoten und/oder Kanten sogenannte Automorphismen. Einfach gesprochen heißt dies, dass sich Knoten und/oder Links in einem symmetrischen Netz gleich verhalten, egal welchen Knoten oder welchen Link man betrachtet. Dies hat äußerst positive Auswirkungen (Vereinfachung) auf die Programmierung, die Lastverteilung und das Routing, da es keine Spezialfälle zu betrachten gibt.

**Skalierbarkeit**

Die Skalierbarkeit gibt das kleinste Netzinkrement (Anzahl von Knoten und Links) an, um dass man eine Topologie erweitern muss, um vertretbaren Aufwand, keine Leistungseinbußen und die Beibehaltung topologietypischer Eigenschaften nach der Erweiterung zu garantieren.

**Konnektivität**

Die Konnektivität gibt die minimale Anzahl von Knoten oder Links (Kanten- bzw. Knotenkonnektivität) an, die durchtrennt werden müssen, damit das Netz als solches nicht mehr funktionstüchtig ist. Sie ist ein Maß für die Anzahl der unabhängigen Wege, die es zwischen zwei verschiedenen Knoten geben kann. Damit beschreibt sie auch die Ausfallsicherheit des Netzes, d.h. je höher die Konnektivität, desto ausfallsicherer ist das Netz.

**Physische Topologie**: beschreibt den Aufbau der Netzverkabelung

**Logische Topologie**: beschreibt den Datenfluss zwischen den Endgeräten d.h. in welcher logischen Beziehung die Geräte beim Datenaustausch zueinanderstehen.

Aus den bis jetzt behandelten Kapitel wissen wir, dass es verschiedene Verbindungsarten zwischen dem Teilnehmen gibt, um Daten zu transportieren. Bei der räumlichen Anordnung eines Netzwerkes unterscheidet.

### Linien-Topologie

Die Linientopologie war einer der ersten Art der Netze, die in kleineren Firmen und privaten Haushalten zu finden war. Ein Koaxialkabel [[5]](#footnote-5)bildet die Basis, die Linie, welches beliebig, bis zu einen gewissen Grad, verlängert werden kann. Die Stationen werden über T-Stücke an das Kabel angeschlossen. Am Anfang und Ende des Kabels sind Widerständeangebracht um Reflexionen [[6]](#footnote-6)zu verhindern.



Abbildung : Linien-Topologie

Abbildung 2 zeigt eine Linien-Topologie mit sieben Geräten. Falls ein Gerät kaputtgeht, wird er Rest des Netzwerkes nicht negativ beeinflusst. Durch Verwendung der Ethernet-Technologie mit der CSMA/CD-Methode hat diese Topologie große Problemen mit Kollisionen. Er kann nur eine Geräte senden.

**Vorteile**

* Der Ausfall eines Teilnehmender hat keine Konsequenzen
* Nur geringe Kosten, da nur geringe Kabelmengen erforderlich sind
* Einfache Verkabelung und Netzerweiterung
* Es werden keine weiteren Geräte zur Übermittlung der Daten benötigt

**Nachteile**

* Alle Daten werden über ein einziges Kabel übertragen
* Datenübertragungen können leicht abgehört werden(Sicherheit-Problem)
* Eine Störung des Übertragungsmediums an einer einzigen Stelle im Bus (defektes Kabel) blockiert den gesamten Netzstrang
* Es kann immer nur eine Station Daten senden. Während der Sendung sind alle anderen blockiert (Datenstau)
* Aufgrund der Möglichkeit der Kollisionen sollte das Medium nur zu ca. 30% ausgelastet werden

### Ring-Topologie

Ein Ring-Topologie ist eigentlich eine Erweiterung der Linien-Topologie, d.h. der erste und die letzten Geräte sind miteinander verbunden. Abbildung 3: Ring-Topologie zeigt ein Ring-Topologie im Einsatz, dieses System ist sehr einfach aufzubauen, aber auch sehr anfällig da der Ausfall eines Gerätes die gesamte weitere Datenübertragung unmöglich macht.

Jede Geräte arbeiten als Repeater [[7]](#footnote-7) , wodurch Netze über große Strecken erzeugt werden können.



Abbildung : Ring-Topologie

**Vorteile**

* Deterministische Rechnernetzkommunikation - Vorgänger und Nachfolger sind definiert
* Alle Stationen arbeiten als Verstärker
* Keine Kollisionen
* Alle Rechner haben gleiche Zugriffsmöglichkeiten
* Garantierte Übertragungsbandbreite
* Skaliert sehr gut, Grad bleibt bei Erweiterung konstant
* Reguläre Topologie, daher leicht programmierbar

**Nachteile**

* Teure Komponenten
* Relativ hoher Durchmesser
* Hoher Verkabelungsaufwand

### Stern-Topologie

Charakteristisches Merkmal der Sterntopologie sind kurze Wege, das bedeutet, dass zwischen Sender und Empfänger nur wenige Vermillungsstationen passiert werden müssen. Abbildung 4 zeigt eine klassische Sterntopologie, diese besteht aus einem zentralen Element (e.g. Switch) und 4 Geräte, die mit einem Ende-zu-Ende Verbindung angeschlossen sind. Fällt ein Gerät aus, stört es die Kommunikation die übrigen Geräte nicht, solange diese nicht mit der Ausgefallenen kommunizieren. Fäll das Switch aus, so können alle Geräte keine Daten mehr austauschen. Dies ist die einzige Schwachstelle dieser Topologie.



Abbildung : Stern-Topologie

**Vorteile**

* Der Ausfall eines Endgerätes hat keine Auswirkung auf den Rest des Netzes.
* Dieses Netz bietet hohe Übertragungsraten, wenn der Netzknoten ein Switch ist.
* Leicht erweiterbar
* Leicht verständlich
* Leichte Fehlersuche
* Kombinierte Telefon- / Rechnernetzverkabelung möglich
* Sehr gute Eignung für Multicast-/Broadcastanwendungen
* Kein Routing benötigt

**Nachteile**

* Durch Ausfall des Verteilers wird Netzverkehr unmöglich
* Niedrige Übertragungsrate bei vielen Geräte, wenn ein Hub benutzt.

### Baum-Topologie

Eine Erweiterung zum Stern stellt der Baum dar (siehe Abbildung 5). Derzeit ist dies meinst genutzte Netzwerktopologie. Prinzipiell wäre eine Verkabelung von jedem Switch zu jedem andern Switch möglich. Man sollte meinen, dass so die kürzesten Strecken erreicht werden.

**Vorteile**

* Der Ausfall eines Endgeräts hat keine Konsequenzen
* Strukturelle Erweiterbarkeit
* Große Entfernungen realisierbar (Kombination)
* Gute Eignung für Such- und Sortieralgorithmen

**Nachteile**

* Bei Ausfall eines Verteilers (Wurzel) ist der ganze davon ausgehende (Unter)Baum des Verteilers "tot".
* Zur Wurzel hin kann es bedingt durch die für Bäume definierte Bisektionsweite von 1 zu Engpässen kommen, da zur Kommunikation von der einen unteren Baumhälfte in die andere Hälfte immer über die Wurzel gegangen werden muss
* Bäume haben mit zunehmender Tiefe (=Anzahl der zu gehenden Links von der Wurzel bis zu einem Blatt) einen sehr hohen Durchmesser. Dies führt in Verbindung mit der Bisektionsweite zu schlechten Latenzeigenschaften bei klassischen Bäumen



Abbildung : Baum-Topologie

### Vermaschtes Netz

Es gibt zwei Arten von vermaschten Topologien. In der ersten, der sogenannten vollständig vermaschten Topologie, weist jeder Knoten eine direkte Verbindung zu jedem anderen Knoten auf. In einer teilweise vermaschten Topologie sind Knoten jeweils nur mit den Knoten verbunden, mit denen sie am meisten interagieren.



Abbildung : das vermaschte Netz zwischen den einzelnen Netzgerät

Nachteile:

* Viele Kabel notwendig
* Sehr hoher Energieverbrauch
* Vergleichsweise komplexes Routing nötig

Vorteile:

* Sicherste Variante eines Netzwerkes
* Bei Ausfall eins Endgerätes ist durch Umleitung die Datenkommunikation weiterhin möglich
* Sehr leistungsfähig

## Netzwerke-Protokolle

In diesem Abschnitt lernen wir die wichtigsten Netzwerke-Protokolle in der industriellen Kommunikation wie beispielsweise PROFIBUS[[8]](#footnote-8), CANOPEN [4], DEVICENET, SERCOS, PROFINET IRT und mehr. Hierbei werden wir hauptsächlich die Protokolle PROFUS und PROFINET IRT betrachten da es des Fokus der Arbeit ist und die entstehende Software ihre Daten nützen soll.

In dem Abschnitt 2.3.1 werden wir untersuchen ganz groß was man unter PROFIBUS versteht und die charakteristische Merkmale einer PROFIBUS-Geräts aussieht und zum Schluss in Abschnitt 2.3.2 werden wir die Grundtheorie der PROFINET IRT untersuchen.

### PROFIBUS

PROFIBUS war einst ein von SIEMENS [5] entwickelter Standard für die Feldbus-Kommunikation in der Automatisierungstechnik. Es ist heute der universelle Feldbus, der breite Anwendung in der industriellen Kommunikation und Gebäudeautomatisierung finden. Es ermöglicht die Verkopplung von Geräte verschiedenen Hersteller ohne besonderes Schnittellenanpassung.

PROFIBUS unterscheidet zwei Gerätetype nämlich:

* **Master-Gerät**: Er ist ein sogenannter „aktiver Teilnehmer“, der den gesamten Datenverkehr auf dem Bus bestimmt. Er ist als einziger befugt, unaufgefordert auf eine Ressource zuzugreifen und Nachrichten ohne externe Aufforderung zu versenden.
* **Slave-Gerät:** Der Slasve ist im Gegenzug „passive Teilnehmer“, welche keine eigenständige Zugriffsberechtigung auf die Ressource habt. Er muss für einen Buszugriff direkt von dem Master aufgefordert werden.

PROFIBUS besitzt drei Protokollvarianten nämlich:

* **PROFIBUS DP** (Dezentrale Peripherie): dient zur Kommunikation zwischen zentralen Automatisierungsgeräten und dezentralen Feldgeräten, durch eine serielle Verbindung. Technische Weiterentwicklungen sind der PROFIBUS-DP/V1 und der PROFIBUS-DP/V2.
* **PROFIBUS PA** (Prozess-Automation): welcher vorrangig im Gebiet der Verfahrenstechnik eingesetzt wird, ist für die Kommunikation zwischen Mess- und Prozessgeräten oder zwischen Aktoren und Prozessleitsystemen zuständig.
* **PROFIBUS FMS** (Fieldbus Message Specification): heute wird diese Variante nach und nach von PROFIBUS DP übernommen und war in der Automatisierungstechnik für komplexe Maschinen und Anlagen vorgesehen.

Die charakteristischen Merkmale eine PROFIBUS-Geräts werden in Form eines elektronischen Gerätedatenblatts die sogenannte GSD-Dateien festgelegt. Anhang dieser Datei werden wir in der Implementierung benötigt, um die Information darzustellen.

Mit dem PROFIBUS kann ein System mit folgende Verfahren realisiert werden:

* reine Master-Slave-System
* reine Master-Master System
* Eine Kombination von beiden Verfahren



Abbildung : Beispielhafte Topologie zur Verdeutlichung der PROFIBUS-System in der industriellen Kommunikation mit einem Master-Slave-Verfahren

Abbildung 6 zeigt eine PROBUS-System im Einsatz mit Master-Salve Verfahren. Hierbei sind 2 POFIBUS-Geräte (PLC und PC) und den Rest sind PROFIBUS-Slave. Die Daten sind sowohl zyklisch als auch azyklisch Prozessdaten von einer Station zu andere Station gesendet. In der Praxis dient die PC zur Konfiguration des Systems und die PLC zur Steuerung des gesamten Systems.

### PROFINET- IRT

# Die Ausgangssituation der bestehenden Softwares

Die Hilscher Gesellschaft für Systemautomation mbH hat das Fachkonzept für die Entwicklung und Produktion von industriellen Kommunikationslösungen für die moderne Fabrikautomation, insbesondere bei der Konfiguration der industrielleren Geräte seit der Tendenz „Industrie 4.0“ [6] stetig fortentwickelt. Aufgrund der Tendenz wurde erstmalig eine Softwareanwendung unter dem Namen „*Communication Studio*“ zur Konfiguration der Geräte entwickelt. Das Fachkonzept wurde erneut fortgeschrieben sowie die Softwareanwendungen für das Web weiterentwickelt. Aufgrund neuer Anforderungen aus dem Bereich Online-Dienstleistungen ergab sich die Notwendigkeit der Erweiterung und Weiterentwicklung der Softwareanwendung. Einer der Erweiterungen nennen wir sie als „*Web-Topology Editor*“.

Die Ist-Aufnahme von „*Communication Studio*“ wurde nach den vorgestellten Techniken aus dem Grundlagenkapitel erarbeitet und für die Erfassung der Informationen hat sich eine Vielzahl von Methoden herausgebildet.

Hierbei werden zunächst die verwendeten Befragungsmethoden für die Ist-Zustand-Analyse kurz vorgestellt und schließlich wird der Anlass zur Erweiterung der Software mit einer Plug-In hinsichtlich der Darstellung der Geräte-Modelle herausgearbeitet und dokumentiert.

## Was ist „Communication Studio“?

„*Communication Studio*“ abgekürzt *CStudio* ist der Überbegriff für verschiedene Softwareprodukte des Unternehmens Hilscher. Die Softwareprodukte werden dabei nach Art der Anwendungen unterteilt:

In dieser Produktfamilie gibt es mehrere WPF-Desktops, die sich nur jeweils im Funktionsumfang unterscheiden. Diese WPF-Desktops wurden mit dem Programmiersprache C# realisiert und bestehen aus vier verschiedenen Hauptprogrammen:

* SolutionExplorer zur Auflistung der feldbusabhängigen Projekte mit bereitgestellte Funktionen
* RibbonView bildete die Toolbox eine weiter eigenständige Anwendung
* PropertyGrid zur spezielle Anzeige bzw. Änderung der spezifischeren Elemente der Anwendung
* Output zur Anzeige von Information wie Debug, Error und Warnung in Laufzeit.

Weiterhin gibt es einige Dienste wie ODM und Device Manager, die die Kommutation mit Hardware ermöglichen.

Interessant ist die Bereitstellung von *Web-Diagnoses* als Auskunftslösung für Konfiguration-Daten. Als weiterer Client für browserbasierte Anwendungen auf HTML-, Angular- oder TypeScript-Basis steht zudem die geplante *TopologyEditor* als Produkt zur Verfügung. Das TopologyEditor wird so implementiert so dass die Netzwerk-Informationen auch direkt Online als Graphen visualisiert werden.

Die Produktfamilie[[9]](#footnote-9) wird aus ersetzender Konfiguration Tool „*Sycon*“ von Firma Hilscher. Aus dieser einzelnen Bausteine können Sie Ihre optimale Konfiguration-Lösung zusammenstellen. Durch *CStudio* können Sie Konfiguration-Funktionalität und -Daten dort anbieten und einbinden, wo sie benötigt werden – am Desktop, via Server oder in Zukunft als mobile Anwendung im Außendienst. Die Ausrichtung von CStudio ermöglicht eine Datenhaltung in dateibasierter Datenquellen, wird aber auch die Verwendung objektrelationalen Datenbank in der Zukunft vorsehen.

Abbildung 8 zeigt die Modularität der Produktfamilie, das modulare Angebot mag für den Interessenten nicht sofort überschaubar sein, bietet aber den entscheidenden Vorteil, dass eine Lösung mit steigenden Anforderungen mitwachsen kann(Skalierbarkeit) und die individuellen Anforderungen zugeschnitten werden kann. Die Gefahr, mit seinem CStudio bezüglich Funktionalität und Anwendungsbereich durchsteigende Anforderungen in einer Sackgasse zu landen, besteht daher nicht.



Abbildung : Übersichtschema *CStudio* mit zukünftiger *Web-Topology Editor*

## Befragungsmethoden

Damit eine umfassende Darstellung des Ist-Zustandes hinsichtlich der Visualisierung der Geräte gewährleistet werden kann, ist es von entscheidenden Bedeutung zuerst eine gute Informationsgrundlage zu schaffen. Eine der wichtigen Informationsquellen sind hierbei die Mittarbeit in meiner Ableitung, da diese die Arbeitsabläufe kennen und über Erfahrungen mit Dokumenten und Daten verfügen. Des Weiteren sind die Dokumente der bestehenden Software, da dort sind die Spezifikationen textuell und graphisch beschrieben.

Die folgenden Methoden werden genutzt, um die Informationen zu erfassen:

**Regelmäßig Meeting**

Bei dieser Methode wird die direkten mündlichen Fragen gestellt. Die Befragung wird entweder auf Basis einen Fragenkatalog durchgeführt oder aber auf Fragen, die im Verlauf des Meetings entwickelt wurden. Darüber hinaus wird das Meeting nicht nur mit dem Projektleiter in Form von Einzel Gesprächen, sondern als Gruppenmeeting durchgeführt.

**Dokumentenanalyse**

Bei dieser Analyse werden Informationen aus Spezifikationen und die Anforderungen der bisherigeren Softwareanwendung „*Communication Studio*“ extrahiert, insbesondere aus der Visualisierung von Geräte-Modelle bei Konfiguration und Überwachung von System.

**Beobachtungen**

Bei Beobachtung werden die Sachverhalte durch sinnliche Wahrnehmung aufgenommen. Da ich schon einige Implementierungen in „*Communication Studio*“ gemacht habe, hat diese Methode viel gebracht um das System zu verstehen.

## Anlass der Plug-In „*Web-Topology Editor*“

Anlass zur Erweiterung von “Communication Studio” mit einer Plug-In sind facettenreich und anspruchsvoll, außerdem ist es uns bewusst, dass die Durführung der Ist-Zustand ist die Grundlage für das Bewerten der Soll-Zustand. Im Folgendes werden die Beschreibung und Darstellung des Ist-Zustands erstellt.

* Das bisherige System bietet eine hierarchische Darstellung der industrieller Devices in „*Solution Explorer*“ (siehe Recht in Abbildung 6): Allerding besitzt das Modell diverse Probleme bei der Erklärung von Datenfluss in der industrielleren Kommunikation
* Die User-Experience um die Topologie in „Communication Studio“ ist unterschiedlich zu handeln, was schwer zu warten da sehr viel unterschiedliche Netzwerke gibt.
* „Communication Studio“ ist nicht Cloud-Fähig. Die Firma Hilscher setzt sich für die neue Technologie
* Die Abwicklung des Projektes bezüglich der Visualisierung der Geräte muss entsprechend dem Client-Server Architektur erfolgen.

# Anforderungsanalyse

Um den Aufwand für Planung und Implementierung einschätzen zu können, ist es eine genaue Analyse der verschiedenen Anforderungen an das System notwendig. Die Analyse muss umfassend und detailliert genug sein, sodass es keine Missverständnisse oder Interpretationsmöglichkeiten während der Entwicklung gibt.

Eine gute Analyse ist der Grundstein für ein realistisches Zeitmanagement und -in großen Projekten - für eine realistische Kosteneinschätzung. [7]

Die folgenden Punkte listen einige Vorteile warum wir überhaupt diese Analyse grundsätzlich durchgeführt haben:

* Kosten senken
* Risiken verkleinern
* Erleichterung beim Erstellen von Testfällen und Testszenarien
* Gute Produkte erleichtern den Verkauf

In diesem Abschnitt werden aus diesen Grund alle Anforderungen an das „Web TopologyEditor“ und das gesamte System detailliert beschrieben, wobei zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden wird. Weitern wird das Use Case Diagramm der Plug-In anhand einiger beispielhafter Use Cases erläutert.

Wir werden zunächst festlegen, was das Softwareprodukt tun soll. Zum Schluss werden wir spezifiziert wie es arbeiten soll.

## Funktionale Anforderungen

Funktionale Anforderungen (FR[[10]](#footnote-10)) spezifizieren, welche Funktionalität oder welches Verhalten das Softwareprodukt unter festgelegten Bedingungen besitzen bzw. erfüllen soll. In einem Wort kann man sich fragen was soll das Softwareprodukt tun?

Hierbei gibt es die Muss-Kriterien, das heißt die Kriterien die auf jeden Fall umgesetzt werden müssen. Sie beschrieben vor allem für den Einsatz des Plug-In essentielle Funktionen. Weiterhin formuliert man die Kann-Kriterien, die umgesetzt werden, wenn es die Resources des Projekts zulassen und die nicht Einsatzfähigkeiten des Systems beeinträchtigen, wenn sie nicht umgesetzt werden können

### Muss*-*Kriterien

*FR010*: Das Plug-In muss einem Nutzer ermöglichen sich mit dem laufenden Projekt ein

Die Darstellung der Topologie in einer Ansicht anzuzeigen

*FR020*: Das Plug-In muss eine Möglichkeit bereitstellen, durch die ein Benutzer die Topologie Darstellung zu schließen.

FR030: Das Plug-In muss einen Benutzer ermöglichen, die Topologie Darstellung des entsprechenden Projekts zu schließen.

FR040: Das Plug-In muss das Editieren von Informationen einer Topologie durch einen Nutzer ermöglichen.

FR050: Das Plug-In muss einen Nutzer die Möglichkeiten bieten, die Topologie wieder zu verwenden.

FR060: Das Plug-In muss Nach der Öffnen die wichtigsten Informationen in einer Ansicht bereitstellen.

FR070: Die Plug-In muss eine graphische Darstellung der Device aus einer Projekt anzeigen.

FR080: Die Plug-In muss dem Nutzer darauf aufmerksam machen, wenn eine Device oder eine Verbindung aus dem Netzwerk ausgezogen wird.

FR090 Das Plug-In muss die Basis-Funktionen Laden, Speichern, Importieren, Exportier von Topologien-Arten bereitstellen.

FR100 Das Plug-In muss die Möglichkeit bieten, die Topologie-arte bezüglich der PROBIFUS und PROFINET-Protokolle darstellen.

FR110 Das Plug-In muss die Möglichkeit bieten das Topologie-Ansicht zu zoomen.

### Kann-Kriterien

FR200: Das Plug-In kann eine „Über Uns“-Ansicht bereitstellen.

FR210: Das Plug-In kann eine Funktion zum Umstellen der Sprache auf English oder chinesisch erhalten.

FR220: Das Plug-In kann eine Funktion zur Umschaltung der Topologie-Arten(siehe Abschnitt 2.2)

FR230 Das Plug-In kann eine Funktion zur Suchen und Ersetzen von Informationen bieten

FR240 Das Plug-In kann eine Funktion zur Filter-Mechanismen bieten.

FR250 Das Plug-In kann die Topologie-Darstellung unabhängig von Protokoll bereitstellen

FR260 Das Plug-In kann eine Möglichkeit bieten, Fehler-Ausgabe darzustellen (A8)

FR270 Das Plug-In kann eine Möglichkeit bieten, Symbole automatisch zu sortieren.

## Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen (NFRs[[11]](#footnote-11)), auch Technische Anforderungen genannt, beschreiben Aspekte, die typischerweise mehrere oder alle funktionalen Anforderungen betreffen bzw. überschneiden (cross-cut). Sie haben in der Regel einen Einfluss auf die gesamte Softwarearchitektur. Außerdem beeinflussen sich nichtfunktionale Anforderungen gegenseitig. Auch hier werden die Anforderungen in Muss-und in Kann Kriterien aufgeteilt.

### Muss-Kriterien

NF010: Es muss möglich sein, die Plug-In zu ändern. Änderungen können Korrekturen, Verbesserungen oder Anpassungen der Software an Änderungen der Umgebung, der Anforderungen und der funktionalen Spezifikationen einschließen. (Wartbarkeit).

NF020: Hier werden die Fähigkeiten des Plug-In, Mängel oder Ursachen von Versagen zu diagnostizieren oder änderungsbedürftige Teile zu identifizieren. (Analysierbarkeit).

NF030: Das Plug-In muss ermöglichen, die Implementierungen einer Spezifikation zu ändern(Änderbarkeit)

NF040 Das Plug-In soll in der Lage seine unerwarteten Wirkungen von Änderung der Software zu vermeiden (Stabilität)

NF050: Das Plug-In muss in der Lage sein, die modifizierte Software zu validieren(Testbarkeit)

NF060: Das Plug-In muss in Communication Studio integriert werden.

NF070: Die gewünschten Informationen müssen in XML- Format gespeichert werden.

NF080: Das Plug-In muss unter der Webbrowsers Google Chrome, Mozilla Firefox und Internet Explorer benutzbar sein.

NF090:

NF100

### Kann-Kriterien

NF110: Das Plug-In kann in Cloud zur Verfügung gestellt werden.

NF120: Das Plug-In kann mit Angular 5 umgesetzt werden.

# Evaluation der Technologien

*TopologyEditor* basiert auf einem verteilten System, das Module zur Umgebungserfassung mit Datenverarbeitungsknoten und Schnittstellen zu andere Programme kombiniert. Der Entwicklungsprozess der Software für ein solches System umfasst neben der eigentlichen Anwendungsentwicklung zwei periphere Aufgaben: Zum einen muss die Kommunikation zwischen Komponenten des Systems sichergestellt und zum andere die Interfaces zur Hardware oder andere externe Programme bereitgestellt werden.

Um die Entwicklungsaufwand an dieser Stelle zu reduzieren muss die beschriebene Vielfältigkeit gekapselt und der Zugriff auf eine uniforme Schnittstelle abgebildet werden. Erst damit ist eine Austauschbarkeit und Wiederverwendbarkeit von Komponenten umsetzbar. Dadurch kann ein der wichtigste Muss-Kriterien aus der Anforderungen abgedeckt werden.

Nach monatelang wurde eine Fülle von verschiedenen Frameworks für die Entwicklung solcher Plug-In gefunden. Nunmehr stehe ich aber vor den Aufgaben aus der Vielzahl an der verfügbaren Implementierung das für meine konkrete Anwendung geeignetste zu finden. Entsprechend muss ich als Entwickler nach dem Abstecken das Szenarienkontexts, wie zum Beispiel der beteiligten Komponente, den nötigen Algorithmen den erforderlichen Kommunikationseigenschaften usw., aufwendige Recherche betrieben, um eine weitegehende Übereinstimmung sicherzustellen. Diese Kapitel dient insbesondere im Clientseitige Umsetzung mit vergleichbare Aufstellung als überblick über den gefundenen Rahmenwerken. Für die serverseitige Umsetzung wird die ASP.NET.Core 2.0 eingesetzt. Anschließend führen wir das Tools bzw. die externen Bibliotheken zur Vereinfachung und Beschleunigung der Entwicklung, zum beispeil für die Visualisierung von Information Daten wir die Bibliothek GoJs genutzt und für das Styling und Strukturierung von Benutzeroberfläche wird das Bootstrap verwendet

## Clientseitige relevante Frameworks

Im den letzten Jahren hat sich im Bereich der Webtechnologie eine Entwicklung von browserfähige Anwendungen die für die Erfüllung spezifischer Aufgabe als eine kompakte Einheit entworfen wurden, hinzu autonomen modulare Systeme vollzogen. Kennzeichnend für diese „neue“ Webtechnologie-Generation ist die Kooperation und Interaktion zwischen Software-Modulen. Um die Integration diese Komponente in einem Gesamtsystem zu ermöglichen und dabei den Software-Entwicklungsprozess zu vereinfachen und zu beschleunigen, sind viele Frameworks mit unterschiedlichen Schwerpunkte entwickelt worden. Diese gehen zum Beispiel Kommunikation und Konfiguration die mit einem modularen System einhergehen, an. In diesem Abschnitt werden nur 3 ausgewählte Konzepte vorstellt. Im Folgende stützt sich dieser Abschnitt auf die wichtigsten Frameworks, die zum einen eine Erfüllung des in Abschb33 beschriebenen Aspekte bieten und zum anderen eine Verbreitung gefunden haben.

### Angular 5

Das Entwickeln und Erstellen von HTML-Seite, die flexibel und einfach zu warten sind, kann eine Herausforderung darstellen. In diesem Abschnitt werden einige der häufigsten Probleme beschrieben, wie bei der Erstellung von HTML auftreten können, und es wird beschrieben, wie Angular-Technologie diese Herausforderungen meisten kann.

In der Regel stehen Entwickler von HTML-Seite vor einigen Herausforderungen. Anwendungsanforderungen können sich im Laufe der Zeit ändern.

Neue Geschäftschancen und –Herausforderungen können sich ergeben, neue Technologie werden möglicherweise verfügbar, oder sogar das kontinuierliche Kundenfeedback während des Entwicklungszyklus kann die Anforderungen der Anwendung erheblich beeinflussen. Daher ist es wichtig, die Anwendung so zu erstellen, dass sie flexibel ist und in Laufe der Zeit leicht geändert oder erweitert werden kann. Entwerfen für dies Art von Flexibilität kann schwer sein zu erreichen denn es erfordert eine Architektur, die es ermöglicht, einzelne Teile der Anwendung unabhängig voneinander zu entwickeln und zu testen, die später isoliert oder aktualisiert werden können, ohne den Rest der Anwendungen zu beinträchtigen

[](file:///C:\Users\GhislainZeleu\AppData\Roaming\Microsoft\Word\Typische%20zusammengesetzte%20Anwendungsarchitektur%20mit%20dem%20Angular%20Framework.PNG)

Abbildung : Typische zusammengesetzte Anwendungsarchitektur mit dem Angular Framework

**Die Wichtigste Konzept von Angular**

Dieser Abschnitt bietet eine kurz überblich über dem wichtigsten Konzept von Angular und definiert einige Begriffe, die in der Dokumentation und in Produktive Code verwendet werden.

* Module: Module sind Pakete mit Services, Components, Direktives und Templates, die unabhängig voneinander entwickelt, getestet, und optional bereitgestellt werden können. In vielen Situationen werden Module von separaten Teams entwickelt und gewartet. Eine typische Angular-Projekt besteht aus mehrere Modulen. Sie können verwendet werden, um bestimmte Geschäftsbezogene Funktionen (z.B. Editieren von Topology, Startseite verwalten) darzustellen und alle Ansichten, Dienste und Datenmodelle zu kapseln, die zum Implementieren dieser Funktionalität erforderlich sind.
* AppModule: In einer Modulare Applikation müssen Module zur Laufzeit von der Hostanwendung erkannt und geladen bzw. importiert werden. In Angular wird ein AppModul verwendet, um anzugeben, welche Module importiert werden sollen und in welcher Reihenfolge.
* Component:
* Template:
* Services:

### Vue.js

### React [8]

## Vergleichskriterien

Nachdem in vorangegangen Anschnitt Beispiele für Clientseitige Framework vorgestellt wurden, widmet sich dieser Abschnitt den Kriterien anhand derer ein systematischer Vergleich durchgeführt werden kann. Der Kriterienkatalog integriert dabei Abschnitt zur Lernkurve und Kosten, fasst die Kommunikationsmechanismen zusammen und einschließlich betrachtet die Programmierkonzepte.

### Lernkurve und Kosten

In der industrielleren Kommunikation integrieren Konfigurationsapplikationen vielfältige protokollabhängige Feldbusse. Um dennoch Applikationen auf Support-Aufkommen bei sowohl Nutzer als auch Entwickler reduzieren zu können, spielen Prinzipien wie Kosten und Lernkurve eine wichtige Rolle.

* **Lernkurve**: stellt die Entwicklung eines Trainierenden beim Erlenen ein Framework dar spiegelt somit den Erfolgsgrad des Lernens über den Verlauf der Zeit. Da diese nicht im Vordergrund in der industriellen Kommunikation sollte diese leicht wie möglich sein.
* **Kosten**: hier sind die negativen Konsequenzen einer erfolgswirksamen Nutzung von einem Framework. Für ein breites Anwendungsspektrum des jeweils Frameworks ist eine Kostenrechnung Lizenzbedingungen und Zielgruppennutzung gekoppelt.

### Kommunikationsmechanismen

Die Infrastruktur umfasst Mechanismen, die eine Interaktion und Kooperation zwischen der einzelnen Komponente des Gesamtsystemes ermöglichen und somit dessen Funktionalität als Ganzes gewährleisten.

* **Kommunikationsprinzip** Das Kommunikationsprinzip beschreibt, in welcher Form der Nachrichtenaustauch zwischen Komponenten erfolgt und in welcher Beziehung die Komponente dabei zueinander stellen. Die Interaktion kann z.B. über eine Event-Kommunikation und über Prozeduraufrufe erfolgen. Als Interaktionsmuster ist das *Publish-Subscribe*-Modell oder *Produce-Consumer*-Modell denkbar.
* **Kommunikationsmiddleware** Damit Anwendungen verteilt über mehrere Rechnerknoten laufen können und somit eine Ortsunabhängigkeit gewährleisten, sind entsprechende Mechanismen erforderlich.

### Programmierkonzept

Diese Kategorie bezieht sich auf Kriterien, die einerseits Eigenschaften der konkreten Implementierung der jeweils Entwicklungsumgebung beschreiben und anderseits die Anwendungsentwicklung eben mit dieser betreffen.

* **Unterstützte Programmiersprachen** Bei der Anwendungsentwicklung sollte

dem Entwickler möglichst die Wahl gelassen werden, in welcher Programmiersprache

entwickelt wird. Eine domain-spezifische Frage zielt dabei auf die Möglichkeit der grafischen Programmierung. Als Programmiersprachen sind JavaScript, C# und TypeScript denkbar.

* **Unterstützungsbibliotheken** Vordefinierte Komponenten z.B. für Verhaltensauswahl Sicherheit, Monitoring und Internationalisierung erleichtern den Entwicklungsprozess und fördern die Wiederverwendung von Software-Modulen, wobei gegebenenfalls entsprechende Anpassungen erforderlich sind.
* **Lizenzbedingung** Der Typ der Lizenz der Frameworks bestimmt insbesondere im Fall der kommerziellen Nutzung über deren generelle Anwendbarkeit. Durch das gewählte Lizenzmodell wird die Breite der Entwicklungs-Community zumindest mitbestimmt. Eine aktive Community erleichtert die Entwicklungsarbeit und bieten in Wikis oder Foren eine Vielzahl von Antworten, Anregungen und Beispielcode

### Zusammenfassung

## GoJS

*GoJS* [9] ist eine JavaScript-Bibliothek, mit der auf einfache Weise interaktive Diagramme in modernen Webbrowsern erstellt werden können. Sie unterstützt grafische Vorlagen und Datenbindung von grafischen Objekteigenschaften zum Modellieren von Daten.

Die Anpassung von Aussehen und Verhalten hängt hauptsächlich von der Einstellung von Eigenschaften ab.

Wir werden in diesem Abschnitt eine kleine Einführung über GoJS bezüglich die Diagramm

Danach die Nachteile Bzw. Vorteile bei Verwendung von dieser externen Bibliothek

### Einführung

Um ein Netzwerktopologie mit Hilfe von GoJS-Diagramm API bereitzustellen, benötigt GoJs ein Modell, das die spezifischen Anwendungsdaten enthält.

Jedes Diagramm besteht aus Knoten, das durch Links verbunden sein könnten und die optional in Gruppen gruppiert sein könnten. Alle Diagramme-Teile sind in Schichte zusammengefasst und nach gewünschten Layouts (z.B. Baum-Layout, Linienlayout …) angeordnet.

Das Diagramm-Modell besteht aus 2 wichtigste Eigenschaften: *nodeDataArray* beinhalten alle Knoten und *linkDataArray* listet alle Verbindungen die in dem Diagramm darstellt werden sollen. Abbildung 12 zeigt die Diagrammklasse und ihre Interaktion: das Ansichtsmodell(Model) implementiert Eigenschaften, an die die Ansicht gebunden werden kann, und benachrichtigt die Ansicht von Statusänderungen über Änderungsbenachrichtigungsereignisse. In der Regel besteht eine Eins-zu-Eins-Beziehung zwischen einer Ansicht(Diagramm) und ihrem Ansichtsmodell(Model)



Abbildung : Die wichtigsten Eigenschaften des Diagramms und ihre Interaktion

Jeder Knoten oder Link wird normalerweise durch eine Vorlage definiert, die sein Aussehen und Verhalten deklariert. Jede Vorlage besteht aus Gruppen von GraphObjects wie TextBlocks (Dient für die Anzeige der Gerätename) Picture (dient für Platzhalter für ein Bild) oder Shapes. Es gibt Standardvorlagen für alle Teile, aber fast alle Anwendungen geben benutzerdefinierte Vorlagen an, um das gewünschte Aussehen und Verhalten zu erreichen. Datenbindungen von GraphObject-Eigenschaften zum Modellieren von Dateneigenschaften machen jeden Knoten oder jede Verknüpfung für die Daten eindeutig.

Die Knoten können manuell (interaktiv oder programmgesteuert) positioniert werden oder können automatisch vom Diagramm.layout angeordnet werden. Knoten besitzen die Möglichkeiten Port als Rechteck darzustellen.

Jedes Diagramm verfügt über eine Reihe von Werkzeugen, die interaktive Aufgaben ausführen, z. B. das Auswählen von Teilen oder das Ziehen von Elementen oder das Zeichnen einer neuen Verknüpfung zwischen zwei Knoten. Der ToolManager [[12]](#footnote-12)bestimmt abhängig von den Mausereignissen und den aktuellen Umständen, welches Werkzeug ausgeführt werden soll.

Jedes Diagramm verfügt außerdem über einen CommandHandler, der verschiedene Befehle implementiert, z. B. Löschen oder Kopieren. Der CommandHandler interpretiert Tastaturereignisse wie z. B. Steuerelement-Z, wenn der ToolManager ausgeführt wird.

Alle programmatischen Änderungen am Diagramm-, GraphObject-, Modell- oder Modelldatenstatus sollten innerhalb einer einzelnen Transaktion pro Benutzeraktion ausgeführt werden, um sicherzustellen, dass die Aktualisierung korrekt ausgeführt wird und um das Rückgängigmachen / Wiederholen zu unterstützen. Alle vordefinierten Tools und Befehle führen Transaktionen aus, sodass jede Benutzeraktion automatisch rückgängig gemacht werden kann, wenn der UndoManager aktiviert ist. DiagramEvents on Diagrams und Event-Handler auf Diagrams und GraphObjects sind alle dokumentiert, egal ob sie innerhalb einer Transaktion ausgelöst werden oder ob Sie eine Transaktion durchführen müssen, um das Modell oder das Diagramm zu ändern.

### Vorteile und Nachteile von GoJS

Aus meiner persönlichen Sicht liegen die wesentlichen Vorteile bei dem Einsatz von GoJS-Diagramm in der breiten Unterstützung der Darstellung von Netzwerkstruktur in Webbrowser.

Man kann an vielen Stellen ein und dasselbe Problem auf unterschiedliche Art und Weise lösen, ohne sich dabei durch die Vorgaben der Dateninformation eingeengt zu fühlen.

Die Dateiinformation lässt also den Softwareentwicklern den Freiraum, nach eigenen Vorstellungen zu modellieren. Besonders Vorteilhaft erscheint die Tatsache, dass GoJs auch die Erweiterungsmöglichkeiten durch TypeScript(mehr dazu Kapitel 9) vorsieht und damit wirklich jedem das Recht anbietet, den Typescript Notationsrahmen zu erschaffen. Diese Möglichkeit sollte aber nur in äußersten Fällen (wenn z.B. keine JavaScript Sprache verwendet soll) verwendet werden, da man sonst der Gefahr, sich vom Standard zu entfernen, entgegenläuft.

Die GoJS nutzt die JSON-Notation für die Konfiguration der verschiedenen Klassendiagramme. Diagramme Konfigurationen können nun nicht nur von den Fachleuten der Software –Firma, sondern auch von den außerhalb stehenden verstanden und verbessert werden.

Weiterer Vorteil liegt in der zunehmenden Verbreitung der GoJS-Bibliothek und ist Open-Source. Die exakten Zahlen sind zwar noch schwer abzuschätzen, es zeichnet sich jedoch ein wachsender Trend für den Einsatz der GoJS-Bibliothek ab.

Die Nachteile lassen sich nun wiederum aus dem Umfang der Bibliothek und Preise ableiten. GoJs ist sehr vielfältig, so dass auch am Anfang sehr viel Aufwand für das Aneignen und Verstehen sämtlicher Konfigurationen aufgebracht werden muss.

Außerdem wird man in der Regel feststellen, dass viele Elemente sehr selten zum Einsatz kommen und damit eher als Ballast der Bibliothek angesehen werden. Um ein Lizenz zu bekommen soll man pro Entwickler über 2500 Euro [10] ausgeben was zu teuer ist.

## Bootstrap

Bootstrap ist ein OpenSource Framework zur Darstellung von Benutzeroberflächen im Browser verwendet

Es erleichtert das Auswahl der Komponenten einer Website und dabei die technischen Anforderungen von verschiedenen Geräten berücksichtigt (Smartphones, Desktops, Tablets) und darüber hinaus gibt das Framework eine solide Grundlage für verschiedene Browseranwendungen.

Die Idee hinter dem Framework ist die Designer und Entwickler auf einen gemeinsamen Nenner zusammenzubringen

Bootstrap wurde mit Augenmerk auf HTML5 und CSS3 entwickelt, um so von den aktuellsten Funktionsweisen profitieren zu können, deutlich wird dies beispielsweise bei den in CSS3 eingeführten Eigenschaften für abgerundete Ecken, Farbverläufe und Schatten.

Bootstrap ist relativ einfach aufgebaut. Sein Kern besteht aus verschiedenen Stylesheets, welche die im Framework vorhandenen Komponenten einbeziehen. Über das zentrale Konfigurationsstylesheet können weitere Anpassungen vorgenommen werden.

* Vorteile: Durch den klaren Leitfaden sparen die Entwickler wie auch die Gestalter Zeit und haben einen gemeinsamen Nenner für die Gestaltung der Oberfläche

Bootstrap bringt von Haus aus Elemente wie Icons, Boxen, Buttons und PullDown Menüs bereits mit

Erweiterungen wie Modal-Boxen, Tooltips und Tabs sind Teil des Frameworks (Integration von JQuery)

Alle Elemente sind gestaltet, wodurch sichergestellt wird, dass jede der möglichen Komponenten einheitlich aussieht

Das Framework ist rückwärtskompatibel. So wird sichergestellt, dass die einzelnen Elemente der Website auf allen Browsern dargestellt werden können

Bootstrap ist anpassbar oder kann durch Erweiterungen mit neuen Funktionen versehen werden

Nachteile

Das komplette Bootstrap Framework ist ein großer Brocken, was die Ladezeit der Website verzögern kann.

Durch die vielen Möglichkeiten verleitet das Framework zum Spielen. Effekte oder falscher Einsatz von Scripts führen eher dazu, dass die Website überladen wird

Das Layout muss sich dem Framework anpassen

Bevor man mit Bootstrap anfängt, sollte man sich mit dem Aufbau des Frameworks befassen und die (meist englische) Dokumentation lesen

## ASP.NET CORE

Bei „*WebTopolgyEditor*“ wird die Interaktion zwischen Client und Server durch den Austausch JSON-basierter Nachrichten geschehen, die mittels http-Protokoll übertragen werden. Da Web Services ein Internetdienst sind, müssen die eingesetzten Technologie Plattformunabhängig und unabhängig von einer bestimmten Programmiersprache sein. REST[[13]](#footnote-13) ist eine Möglichkeit um Web Services zu implementieren und da wir die C#-Programmiersprache gewählt haben, werden wir als Framework ASP.NET Core 2.0 von Microsoft nehmen.

### REST

„REST“ ist plattform-und programmiersprachenunabhängig, und ist einer der verbreiste Möglichkeit um Web Services zu realisieren, er ist die Architekturvorbild für das Internet und ist geeignet für die Erstellung von Web Services, und ist kein Standard wie SOAP[[14]](#footnote-14) .

Ein REST-System besteht aus Resources die per URI[[15]](#footnote-15) adressiert werden (z.B. Ein Device in einer industrielleren Netzwerksystem, über eine URI adressiert und kann so angesprochen werden)

Rest besitzt folgende Merkmale

* Die gesamte Nachricht wird in URL kodiert.
* Jede Anfrage muss alle notwendigen Informationen für die Durchführung beinhalten (da http stateless ist)
* Darstellungen werden untereinander verlinkt, um dem Client die Möglichkeit zu geben, von einem Zustand in den nächsten zu wechseln.
* Weder Client noch Server müssen die Bedeutung einer URI verstehen
* Ändert sich die Darstellung einer Ressource oder werden neue Ressourcen zur Verfügung gestellt, kann das Interface des Clients beibehalten werden.
* Konsumiert die http-Methoden wie GET, POST, PUT und DELETE

### ASP.NET Core 2.0

ASP.NET Core 2.0 ist ein plattformübergreifendes, leistungsstarkes Open-Source-Framework zum Erstellen moderner, cloudbasierter mit dem Internet verbundener Anwendungen.

Bei ASP.NET Core 2.0 handelt es sich um eine Neugestaltung Framework von Microsoft mit der Architektur, die ein schlankeres Framework mit größerer Modularität ergeben.

ASP.NET Core bietet die folgenden Vorteile:

* Eine einheitliche Umgebung zum Erstellen der Webbenutzeroberfläche und von Web-APIs
* Integration von modernen clientseitigen Frameworks und Entwicklungsworkflows
* Eine schlanke, leistungsstarke und modulare HTTP-Anforderungspipeline
* Fähigkeit zur Erstellung und Ausführung unter Windows, macOS und Linux
* Open Source und mit Fokus auf der Community
* ASP.NET Core besteht vollständig aus NuGet-Paketen.

# Grundprinzipien von *WebTopologyEditor*

In diesem Abschnitt wird zuerst die FDT Technologie eingeführt und werden grundlegende Konzepte für den TopologyEditor in CommunicationStudio vorgestellt, danach werden Kernszenarien dargestellt um zu zeigen wie die Datenquelle erzeugt wird. einschließlich der Kommunikation zwischen TopologyEditor Client und TopologyEditor Server. wobei deren Gestaltung in Beziehung zu den in Abschnitt **Fehler! Verweisquelle konnte nicht gefunden werden.** behandelten Konzepten gesetzt wird.

## FDT

FDT ist eine Technologie, die den Datenaustausch zwischen Feldgeräten und Automatisierungssystemen unterstützt. Die Technologie basiert auf einer Schnittstellenspezifikation, die als IEC 62453 standardisiert ist. Die Spezifikation definiert zwei Hauptkonzepte: Device Type Manager (DTM) und Frame Applikation. Ein DTM ist eine Softwarekomponente, die für einen Feldgerätetyp spezifisch ist. Eine Rahmenanwendung ist eine Softwareumgebung (Teil des Automatisierungssystems) zur Integration von DTMs. Innerhalb einer Rahmenapplikation stellt jeder DTM spezifische Daten und Dienste für das jeweilige Feldgerät zur Verfügung. Da die Technologie auf einer standardisierten Schnittstelle basiert, kann jeder DTM in jede Rahmenanwendung integriert werden. Basierend auf FDT ist es möglich, Kommunikationsgeräte, Kommunikationsinfrastrukturgeräte (z. B. Gateways) und Feldgeräte abhängig von ihren Kommunikationsprotokollen zu integrieren. Die Unterstützung verschiedener Kommunikationsprotokolle wird durch zusätzliche Kommunikationsprotokollspezifikationen z. B. für PROFINET und PROFIBUS (siehe Abschnitt 3.4), Ethernet IP, DeviceNET, HART und CanOPen) oder durch herstellerspezifische Protokollintegration wie Beispielerweise Hilscher-Protokoll bereitgestellt. Die aktuelle Version ist FDT2.0.

## TopologyEditor grundlegende Konzepte

*WebTopologyEditor* ist nicht Teil des FDT2.0-Projekts. In diesem Abschnitt werden die grundlegende Konzepte von dem Plug-In erklärt und Leser zeigen, wie die zukünftige *TopologyEditor-Plug-In* implementiert werden könnte. Eine Rahmenanwendung wie *CommunicationStudio* kann TopologyEditor unterstützen, um die Datenquelle zu erzeugen.

*WebTopologyEditor-Plug-In* besteht aus zwei Hauptsoftwarekomponenten nämlich Der *WebTopologyEditor-Server* und *Web-Topologie-Editor-Client*.

Der Hauptanwendungsfall für *WebTopologyEditor* ist der Offline/Online-Datenaustausch zwischen Geräten und browserfähige Dokumente mit Data-Access-Funktionalität um die Topologie des entsprechenden Systems darzustellen. In diesem Anwendungsfall werden die Gerätedaten von einem *WebTopologyEditor*-Server bereitgestellt und von einem *WebTopologyEditor*-Client konsumiert, der in das browserfähige Dokument integriert ist.

*WebTopologyEditor* bietet Funktionen zum Darstellen eines physikalischen Anordnung (Verknüpfungen falls es zulässt) der protokollabhängige Geräte mit Datenelementen sowie zum Lesen, Importieren, Exportieren, Löschen, Schreiben und Überwachen dieser Elemente für Datenänderungen.

Die Spezifikationen von *WebTopologyEditor*-Server basieren auf der Microsoft ASP:NET Technologie und von *WebTopologyEditor* –Client auf dem Google Angular –Technologie für die Kommunikation zwischen Softwarekomponenten verschiedener Hersteller. Daher sind der *WebTopologyEditor* -Server und die Clients nicht nur auf Windows-basierte Automatisierungssysteme beschränkt aber auch auf Web-Anwendung möglich

*WebTopologyEditor* enthält alle Funktionen der DtmWebApi-Klassenspezifikationen (mehr dazu Abschnitt 5.2) definiert jedoch plattformunabhängige Kommunikationsmechanismen sowie generische, erweiterbare und objektorientierte Modellierungsfunktionen für die Informationen, die ein System bereitstellen möchte.

Der *WebTopologyEditor* -Netzwerkkommunikationsteil definiert Mechanismen, die für verschiedene Anwendungsfälle optimiert sind.

Die entstehende Version von *WebTopologyEditor* wird ein optimiertes HTTP-Protokoll für hochperformante Intranet-Kommunikation sowie eine Abbildung auf akzeptierte Internetstandards wie Web Services definiert. Das abstrakte Kommunikationsmodell hängt nicht von einer spezifischen Protokollzuordnung ab und ermöglicht das Hinzufügen neuer Protokolle in der Zukunft. Funktionen wie Sicherheit, Zugriffskontrolle und Zuverlässigkeit sind direkt in die Transportmechanismen integriert.

Das *WebTopologyEditor* -Modell bietet eine Standardmethode für Server, um Objekte Clients zur Verfügung zu stellen. Objekte können sich aus anderen Objekten, Variablen und Methoden zusammensetzen. *WebTopologyEditor* erlaubt auch, Beziehungen zu anderen Objekten auszudrücken.

Die Menge der Objekte und zugehörige Informationen, die ein *WebTopologyEditor* Server Clients zur Verfügung stellt, werden hier als Topology bezeichnet. Die Elemente des *WebTopologyEditor* -Objektmodells werden im Topology als eine Gruppe von Knoten dargestellt, die durch Eigenschaften beschrieben und durch Referenzen miteinander verbunden sind.

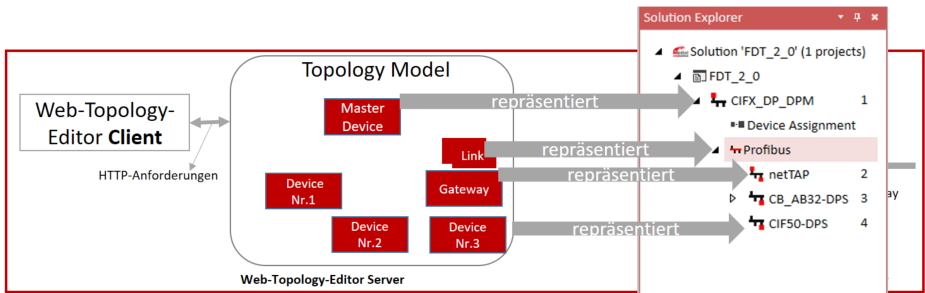
*WebTopologyEditor* definiert drei Klassen von Knoten zur Darstellung von Topology -Komponenten. Die Klassen sind Objekte, die Geräte, Controller und Kommunikationsinfrastrukturgeräte repräsentieren. Jede Knoten-Klassen und Verbindung-Klassen haben einen definierten Satz von Eigenschaften und Methoden.

In einem Multiclient-Szenario ist die *CommunicationStudio* für die Verwaltung des Zugriffs auf die DTMs zuständig.

## Kernszenarien

In diesem Abschnitt werden die häufigsten Szenarien beschrieben, die bei der Arbeit mit TopologyEditor in CommunicationStudio auftreten. Diese Szenarien umfassen das Offline-Konfiguration eines Projekts, das Registrieren und Erkennen von Geräte, das Laden von XML-Topologie, das Erkennen, wenn ein Device eingefügt bzw. gelöscht wurde. Sie können im Offline-Module ein gerätspezifischer Device in einer FDT2.0-Projekt oder durch Scannen eines lokalen Verzeichnisses registrieren und erkennen.

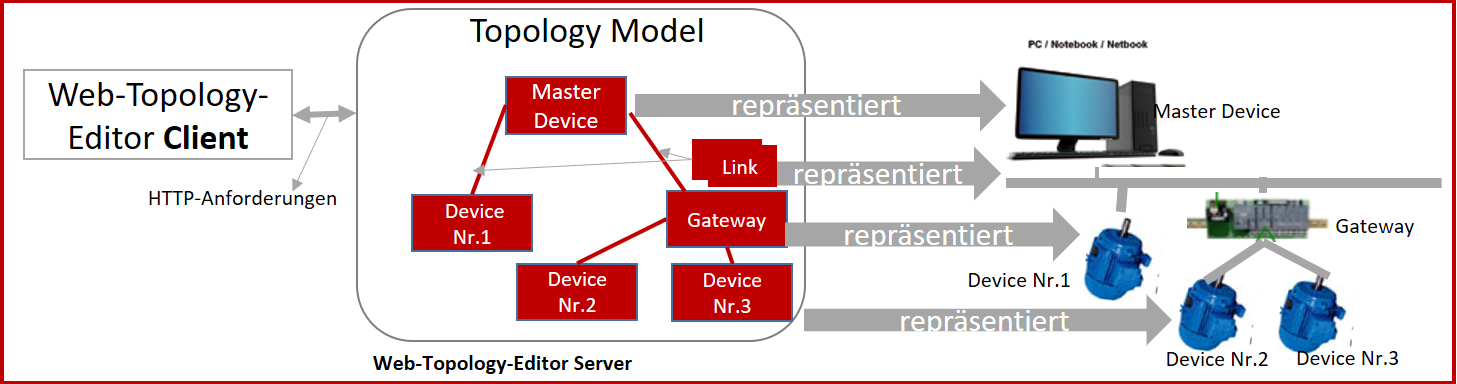
### Offline Szenario



### Online Szenario

In Laufe der Applikation wird durch einen sogenannten Scan-Mechanismus der spezifischen Anordnung der Geräte, die untereinander verbunden sind, generiert.

Die generierten Daten werden als JSON-Datei dargestellt. Der Zweck einer JSON-Datei besteht darin ein Diagramm zu definieren, das ein Netzwerk darstellt.



## Datendarstellung (Prozessdaten, Bedienungsdaten)

Das Konzept des TopologyEditor soll prinzipiell die physikalische Netzstruktur einer industriellen Kommunikation auf einer Schnittstelle (Laptop Mobile Geräte) bereitstellen.

Um grundlegende Konzepte erklären zu können, betrachten wir in dem folgenden Abschnitt eine Mini-Topologie, die nur aus 2 Knoten und eine Kante besteht.

Es enthält 2 Knoten und eine Kante: jeden Knoten repräsentiert ein Device und eine Kante eine spezifische Verbindung. Die Mini-Topologie wird durch die folgende JSON-Datei repräsentiert



Abbildung : Die Output JSON-Dateien für die Mini-Topologie

Das Output des Scan Mechanismus besteht aus einen Root Element und einer Vielzahl von Unterelemente: Graph, Knoten, Kante. Im weiteren Verlauf dieses Abschnitts werden wir diese Elemente im Detail besprechen und zeigen, wie sie ein Diagramm definieren.

Die Topologie-Struktur, die Knoten und Kanten enthält, wird als Graph bezeichnet. Ein Graph besteht aus zwei JSON Dateien nämlich:

* Geräteinformationen: In einem Geräteinformation JSON-Datei sind die Deklarationen von Knoten mit dem Schlüsselwort „*devices*“ und Eigenschaften der Topologie als Root-Elemente definiert. Die Liste von Ports von jeweils Geräte wird innerhalb eines –device-Element mit dem Schlüsselwort „Ports“ definiert. und
* die Verbindungsinformation: hier wird die List von Verbindungen mit dem Schlüsselwort „links“ definiert und jede link-Element repräsentiert eine Verbindung zwischen 2 Geräten. Durch das Schlüsselwort „*from*“ wird das Quellgerät identifiziert und „to“ das Zielgerät identifiziert bzw. „formPort“ wird das Port des Quellgerät und „toPort“ Zielgerät festgelegt.

Knoten repräsentieren die Geräte im Graphen und werden mit Hilfe vom Schlüsselwort „devices“ aufgelistet Jeder Knoten hat eine Kennung, die innerhalb des gesamten Dokuments eindeutig sein muss, d. H. In einem Dokument dürfen keine zwei Knoten mit der gleichen Kennung vorhanden sein. Die Kennung eines Knotens wird durch „*systemTag*“-Schlüsselwort definiert, die eine Zeichenfolge ist. Jeder Knoten muss ein „displayName“-Wert haben, der den Namen von der Device bereitstellt (Siehe Abbildung 10) und „stationAddress“-Wert haben der die Gerätadresse definiert.

Die andere Option können optional sein wie (Ports, Bilder, Range ...)

Die Verbindungen werden durch Kanten im Graphen repräsentiert. Jede Kante muss mit Hilfe der „link“-Schlusswort aufgelistet werden. Sie muss ihre zwei Endpunkte mit der Quelle und dem Ziel definiert sein. Der Wert der Quelle bzw. Ziel muss der „*SystemTag“* eines Knoten sein. Genauer so soll der Wert der Quelle-Port bzw. Ziel-Port der „*portId*“ eines Knoten sein

Jede Kante kann ein optionales Schussel-Label haben, das eine Zeichenfolge ist.

Das Gewicht der Kante wird durch das optionale Schlüsselwort Gewicht festgelegt und ist ein Float.

## Zukunftsaspekte (evtl. Erweiterungen)

Da die Physikalische Topologie ein wachsendes Thema der Forschung in der industriellen Kommunikation ist, werden wir in der Zukunft ein eigenes Format definieren, für das Output des Scan-Mechanismus die sogenannte TOPO Dokument.

Das TOPO-Dokument wird ein XML-Dokument basiert sein.

In der Spezifikation der TOPO-Dokument soll folgendes definiert sein:

* Wie ein Graph mit Hilfe von XML-Elemente und XML- Attributes definieret werden soll
* Wie Knoten und Verbindungen mit Hilfe von XML-Elemente deklariert werden sollen.
* Das Dokument soll als TOPO Datei gespeichert werden.
* TOPO soll die XML-Schemadatentypen XSD 1.1 für die Grundelemente wie ganze Zahl, Datum, Zeichenkette ... verwenden

Es wird auch eines Mechanismus für Übertragungssicherung in die Zukunft geplant

Wie schon in der Einleitung erwärmt wird, wird auch Echtzeitverhalten während der Kommunikation zwischen Komponente geplant.

Für die Zulässigkeit einer Topologie Arte wird auch eine eigene Module geplant.

# Entwurf

In diesem Kapitel wird das zu realisierende Plug-In entworfen. Hierfür wird zunächst auf die Architektur *TopologyEditor* eingegangen und anschließend das Datenmodell erläutern. Ausgehend von den erhobenen funktionalen und nicht funktionalen Anforderungen in Abschnitt 7.1 und 7.2 entsteht der Entwurf.

## Architektur der *TopologyEditor*

Die Architektur der TopologyEditor basiert auf die Architektur einer SPA[[16]](#footnote-16) denn der SPA-Ansatz reduziert die Zeit die von der Anwendung für die Reaktion auf Benutzeraktion , was eine mehr flüssige Erfahrung ist.

In herkömmlichen Webanwendung verarbeitet der Server die Anforderung und sendet Sie im Browser als Reaktion auf die neue Aktionsanforderung von Client, jedoch bei SPA muss der Browser nur der Bereich auf die Seite zu aktualisieren, die geändert wurden; Es ist nicht erforderlich, die gesamte Seite erneut zu laden.

Die Architektur von TopologyEditor umfasst einige Herausforderungen, die nicht in herkömmlichen Webanwendung vorhanden sind. Allerdings begegnet Technologien wie ASP:NET:Web-API, JavaScript-Framewort wie Angular 5(Abschnitt 6.1) und CSS-Framework wie Bootstrap(Abschnitt 6.2) können entworfen werden und Aufbauen von SPAs wirklich vereinfachen.



Abbildung : Diagramm zeigt den Grundlegenden Entwurf der TopologyEditor

Die Abbildung zeigt die den Grundentwurf der TopologyEditor und wird im Folgenden erklärt.

Der Client ist das Medium, das die Web-API nutzt (mobile App, Browser usw.). Hier wird der Client als Angular App erstellt. Das Angular App besteht aus 4 Hauptmodulen und werden als Typescript-Klassen dargestellt.

Die Modelle sind Objekte, die die Daten in der App darstellen. In diesem Fall sind drei Hauptklassen erstellt Modelle werden als C#-Klassen dargestellt, die auch als Plain Old C# Object (POCOs) bezeichnet werden.

Controller sind Objekte, die HTTP-Anforderungen verarbeitet und die HTTP-Antwort erstellt. Sobald eine Anfrage des Clients mit der URI eingeht, wird die verlinkte Funktion in dem betreffenden Controller aufgerufen. Die Controller repräsentieren die Anwendungslogik der Applikation. Sie können mit den Modellen und Wrapper (DtmAPI) interagieren, um insbesondere Daten abzufragen.

Den Wrapper DtmAPI ist die Schnittstelle für die feldbusprotokollabhängigen Parameter und die Einstellungen die hier an der Stelle nicht betrachtet werden sonders als gegeben vorausgesetzt werden.

## Datenmodell

Aus Gründen der Übersicht wird in diesem Abschnitt nur auf die für die Plug-In besonderes relevanten Entitäten der TopologyEditor eingegangen (Abbildung 13). Eine vollständige Übersicht befindet sich im angehängte CD.

Aus den funktionalen Anforderungen ergibt sich direkt das Domänenmodell.



Abbildung : Domänenmodell der Elemente des TopologyEditor

Das Entität *Topology* repräsentiert die Topologie des bereitgestellten industriellen Netzwerks ohne Berücksichtigung der Verbindungen und enthält wesentliche Merkmale wie etwa die Überschrift *Title,* Die Identifikationsnummer *TopologyId* und vor allen die Liste der Geräte die sich in Netzwerk befinden.

Jede Device kann wiederum mehrere Ports besitzen; Ein Port entspricht den Zugangspunkt einer Device und wird von dem Entität *Link* verwenden um die Verbindung herzustellen

Das Entität *LinkData* ist verantwortlich für die Verbindungen einer Topologie zu listen und die dazugehörige Topologie zu identifizieren

## DtmApi Wrapper

DTMAPi besteht aus zwei Hauptfassaden nämlich TopologyFassade und LinkFassade, die Methoden und Eigenschaften bereitstellt, die eine Untermenge der Funktionalität des Systems darstellen. Andere Klassen greifen nur noch auf diese Fassaden zu. Dadurch wird die Benutzung der Gruppen von Klassen (sind meisten feldbusprotokollabhängig) einfacher und Veränderungen hinter den Fassaden bleibt ohne Auswirkungen auf die Nutzer der Fassaden. Als Nebeneffekt wird die Funktionalität für die Entwickler leicht zu verstehen, da sie nur noch die Fassaden und nicht die Details der Gruppe der feldbusprotokollabhängigen Klassen verstehen müssen. Man kann sagen, dass diese Fassaden die Abstraktion der Gruppe darstellen.



Abbildung : Klassendiagramm eines Beispiel für Fassaden mit feldbusabhängigen Protokollen

Abbildung 14 zeigt Controller, die Fassaden als Beobachter für die Tätigkeiten wie (zum Beispiel die Add, Remove und Change Device) angemeldet sind. An welchen Konkreten Feldbusklassen sie sich tatsächlich anmelden oder wie dieses Feldbusse intern aufgebaut sind, spielen so für Controller keine Rolle.

Auf Grund der Übersicht wurden die Fassadenklassen sichtbar mit nur jeweils eine Methode gemacht (schwarz fett), um die wichtige Rolle der Wrapper zu erklären.

Die Plug-Ins-Feldbusse sind Runde Rechtecke mit rote Rahmen.

## Ansicht der TopologyEditor

# Realisierung

In diesem Kapitel wird kurz auf die Punkte der Implementierung des Plug-In „*WebTopologyEditor*“ eingegangen. Hierzu gehört die Auswahl der verschiedenen Programmiersprachen und die verwendeten Werkzeuge

## Vorgehensweise

Beim Unternehmen Hilscher Gesellschaft ist die *Scrum[[17]](#footnote-17)* Vorgehensmodell in der Abteilung User Interface zur Durchführung von Softwareprojekten vorgeschrieben. Da es sich um eine Plug-In handelt, wird ein iterativ inkrementelles Vorgehensmodell eingesetzt. Dadurch kann man das implementiere Module schrittweise in des bestehen Software integriert kann.

## Verwendete Werkzeugen

In diesem Kapitel werden wichtige bei der Umsetzung verwendete Technologien und Werkzeuge angesprochen. Auch nichtfunktionale Punkte wie die Wiederverwendbarkeit des QuellCodes oder bereits eingesetzte Technologien in der Firma spielen eine Rolle. Für die Realisierung des Plug-In „*WebTopologyEditor“* wurde als grundlegendes Werkzeug die Entwicklungsumgebung Microsoft Visual Studio 2017 einerseits für Serverseitige Implementierung und als Programmiersprache C# gewählt und andererseits (clientseitige Realisierung) Visual Studio Code. Des Weiteren werden Bootstrap und ASP:NET Core verwendet, wie bereits in Kapitel 6 erläutert

## Ausgewählte Implementierungsaspekte

In diesem Kapitel werden ausgewählte Teile der Implementierung vorgestellt. Hierbei stehen die Erfassung der Nutzereingaben und die daraus resultierenden Verarbeitungsschritte dieser Daten bis zur Anzeige der Topologie im Vordergrund. Zunächst wird in Kapitel 8.3.3 darauf eingegangen, wie verschiedene Geräten- und Verbindungsinformationen übertragen werden.

Anschließend wird in Kapitel 8.3.4 gezeigt, wie anhand dieser Informationen eine Topologie Aktualisiert wird und geeignete Daten gefunden werden. Im Kapitel 3 wird betrachtet, wie mit Hilfe der gefundenen Daten die Topologie erstellt wird. Im Folgenden Kapitel 4 steht die Darstellung der fehlerbehafteten Statistik im Vordergrund. Abschließend wird in Kapitel 5 die graﬁsche Umsetzung der Darstellung und der Auswertung erläutert.

### Serverseitige Kodierung

Der Server ist ASP.Net Core basiert mit der Vorlage Web-API. Er besteht aus drei wesentlichen Komponenten nämlich Hilscher.TopologyEditor und Hilscher.TopologyEditor.AspNet und natürlich die Hilscher.Web.API

Die TopologyEditor Library hilft die Software „Communication Studio“ alle Netzwerk Protokolle für Web-Services mithilfe des Model-View-Controller-Musters zu kommunizieren. Diese Library unterstützt diejenigen, die mit ASP.NET Core eine Reihe von Anwendungsstilen erstellen möchten. Sie wurde jedoch momentan hauptsächlich für PROFINET IRT entwickelt, die aus einzelnen, funktional vollständigen Teilen bestehen, die zusammen eine einzelne integrierte Schnittstelle bilden. Die TopologyEditor Library beschleunigt die Entwicklung von Anwendungen mit bewährten Entwurfsmustern.



Abbildung : Zusammengesetzte Anwendungsarchitektur mit häufig verwendeten Paketen

**Hilscher.TopologyEditor** enthält die Kernfunktionalität von TopologyEditor, die auf unterstützte Software Communication Studio genutzt wird, and um die feldbusprotokollabhängige Parameter zu interagieren. Das beinhaltet:

**Modellklassen**: wie Node, Link, Toplogy die unterandere die spezifische DTM und Einstellung repräsentiert

**Events** stellt die Ereignishandler bereit, die das Auftreten einer Aktion wie „Das Laden von Parameter beenden“ signalisiert.

**Interfaces**: stellt die Schnittstellen zur Kommunikation mit physikalischen Geräten beziehungsweise DTM-Geräten bereit. Zu dieser Schnittstelle gehören beispielerweise IDataAccess, IDtmDataAccess. Außerdem definiert sie die Schnittstellen INodeManger und ILinkManger zur Verwaltung von Node und Link

**Hilscher.TopologyEditor.AspNet** stellt Klassenkomponente bereit, um ASP.NET mit TopologyEditor Library zu verwenden. Zu diesen Komponenten gehört die tatsächliche Implementierung von allen Schnittstellen von Interfaces und viele Hilfeklassen

**Facades** sind für die Konkrete Klassen zur Kommunikation mit DTM-Geräte verantwortlich, darin befindet sich Beispiel TopologyDataAccess

Managers: beinhaltet die konkreten Implementierungen von Mangers-Schnittstellen aus Interfaces. Dazu gehören NodeManger, LinkManager

**Utils** sind meisten die statischen Klassen zur Erleichtern beim Erstellen von Modellkassen und definiert statische schreibgeschützte Eigenschaften um spezifische Parameter beim Kodierung freizugeben

**Hilscher.TopologyEditor.Web.API** ist eine Sammlung von Controller, die das HTTP-Anforderungen verarbeitet und die HTTP-Antwort erstellt. Sie Konsumiert die zwei vorgestellte Assemblies. Diese Komponente hat zwei Controller, einer der die Device-Anfragen verarbeitet und andere die Verbindungen-Anfragen bearbeiten.

### Clientseitige Kodierung

### Darstellung der Topologie anzeigen

### Darstellung der Topologie aktualisieren

### Darstellung der Topologie speichern und laden

### Fehlerbehandlung einer Topologie

# Systemtest

Es handelt sich hier um einen reinen Black-Box-Test, in dem die Art oder Details der Implementierung nicht berücksichtigt werden. Dieser Abschnitt beschreibt Testfälle der auf den Anforderungen basiert, ausführlich.

User Story: Eine Anwenderin initialisiert das Web-Topology-Edtor. Das System scann automatisch das vorgelegte Netzwerk (IRT Proktokolle) und eine Darstellung der Geräte wird angezeigt. Sie verschiebt die Position einer Device und löscht eine Verbindung. Sie speichert abschließend das Diagramm, um es später wieder zu laden.

In der User Story lassen sich folgende Anwendungsfälle identifizieren:

1. Das Plug-In initialisieren
2. Netzwerk scannen
3. Diagramm anzeigen
4. Device verschieben
5. Verbindung löschen
6. Diagramm speichern
7. Diagramm laden

Die folgenden Anwendungsfälle beschreiben, wie Basisfunktionen von der „*WebTopologyEditor*“-Systemarchitektur unterstützt werden.

## Anwendungsfall: Topologie anzeigen

Der „*WebTopologyEditor*“-Client zeigt dem Benutzer die Topologie in einer hierarchischen Baumstruktur an. Der Baum wird von einem „*WebTopologyEditor*“-Client für das Topology Model bereitgestellt. Wenn der „*WebTopologyEditor“*-Server eine Liste von Verbindungen liefert, wird diese gemäß dem in „*WebTopologyEditor*“–Client definierten "Link Model" bereitgestellt.

|  |  |
| --- | --- |
| Use Case | Toplogie anzeigen |
| Stakeholder |  |
| Preconditions |  |
| Post Conditions |  |
| Actors |  |
| Trigger |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Anwendungsfall: Gerät identifizieren

## Anwendungsfall: Gerät Parameter lesen

## Anwendungsfall: Geräte manuell verbinden

## Anwendungsfall: Verbindung Parameter lesen

## Anwendungsfall: Topologie exportieren

## Anwendungsfall: Topologie importieren

# Zusammenfassung und Ausblick

## Zusammenfassung

## Ausblick

Während dieser Arbeit

Literatur

|  |  |
| --- | --- |
| [1] | S. Kersken, IT-Handbuch für Fachinformatiker, Bonn: Galileo Press, 2013. |
| [2] | K.-U. W. Gottfried Vossen, Grundkurs Theoretische Informatik, Trier: Vieweg+Teubner Verlag , 2016. |
| [3] | M. Popp, Industrielle Kommunikation mit PROFINET, Karlsruhe: ProfiBus Nutzerorganisation e.V.(PNO), 2016. |
| [4] | K. GMBH, „KUNBUS,“ , 01 2016. [Online]. Available: https://www.kunbus.de/canopen-grundlagen.html. [Zugriff am 01 03 2018]. |
| [5] | S. AG, „SIEMENS,“ 01 2018. [Online]. Available: http://w3.siemens.com/mcms/automation/de/industrielle-kommunikation/profibus/Seiten/Default.aspx. [Zugriff am 01 03 2018]. |
| [6] | T. M. Heinrich Hippenmeyer, Automatische Identifikation für Industrie 4.0, Berlin, Heidelberg: Springer Vieweg, 2016. |
| [7] | I. Sommerville, IT Informatik Software Engineering, Pearson, 2007. |
| [8] | Facebook OpenSource, „React,“ Facebook Inc., 2018. [Online]. Available: https://reactjs.org/docs/introducing-jsx.html. [Zugriff am 01 03 2018]. |
| [9] | Northwoods Software, „GoJS,“ 01 01 1998. [Online]. Available: https://gojs.net/latest/index.html. [Zugriff am 04 04 2018]. |
| [10] | „Northwoods Software,“ GOJS, 01 01 1998. [Online]. Available: https://www.nwoods.com/store/p-82-gojs-oem.aspx. [Zugriff am 05 04 2018]. |
| [11] | Hilscher GmbH, „Portal Hilscher,“ 01 10 2017. [Online]. Available: https://kb.hilscher.com/display/COMSTUDIO/2017-06-02+Topology+in+Communication+Studio. |

Index

A

Abbildung 5

Abschnitt 5

B

Beispiel 7

Beweis 7

D

Definition 7

E

Echtzeiten 9

F

Formel 6

Freigabekonsitenz 8

I

Inkonsitenz 8

K

Kapitel 5

Konsitenz 8

schwach 8

Konsitenzmodelle 8

L

Lemma 7

Linearisierbarkeit 9

Literaturverweise 7

M

Matrix 6

O

Operation

Lese 9

Schreib 9

Optimierung 8

Q

Quelltext 5

S

Satz 7

schwach 8

sequentiell

Konsitenz 8

streng 8

T

Tabelle 5

U

Unterkapitel 5

V

Vektor 7

Verfügbarkeit 8

Glossar

|  |  |
| --- | --- |
| DisASTer | DisASTer (Distributed Algorithms Simulation Terrain) A platform for the Implementation of Distributed Algorithms |
| DSM | Distributed Shared Memory |
| AC | Linearisierbarkeit (atomic consistency) |
| SC | Sequentielle Konsistenz (sequential consistency) |
| WC | Schwache Konsistenz (weak consistency) |
| RC | Freigabekonsistenz (release consistency) |

Erklärung der Kandidatin / des Kandidaten

* Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.
* Die Arbeit wurde als Gruppenarbeit angefertigt. Meine eigene Leistung ist im Kapitel „Verantwortliche“ zu Beginn der Dokumentation aufgeführt.

Diesen Teil habe ich selbstständig verfasst und keine anderen als die angegebenen   
 Quellen und Hilfsmittel verwendet.

Name der Mitverfasser: ....................................................................................................  
 ...........................................................................................................................................

|  |  |  |
| --- | --- | --- |
| Datum |  | Unterschrift der Kandidatin / des Kandidaten |

1. *Plugin* oder Add-on ist ein Softwaremodul in der Softwaretechnik, das eine bestehende Software erweitert, um spezifische Aufgaben zu erledigen. [↑](#footnote-ref-1)
2. *IRT* steht für isochronous realtime [↑](#footnote-ref-2)
3. *Performance* kennzeichnet die Leistungsfähigkeit eines Netzwerkes und dessen Komponenten [↑](#footnote-ref-3)
4. nennt man in Netzwerk den Weg von einem Netzknoten zum nächsten [↑](#footnote-ref-4)
5. Kupferkabel mit einem konzentrischen, elektrischen Leiter, der durch eine Isolierung in einem festen, mittigen Abstand zur Abschirmung gehalten wird [↑](#footnote-ref-5)
6. *Reflektionen* sind Wellen, die sich in beiden Richtungen auf der Übertragungsleitung fortpflanzen. So wie ein Echo. [↑](#footnote-ref-6)
7. Verstärker für das empfangene Signal [↑](#footnote-ref-7)
8. Process Field Bus [↑](#footnote-ref-8)
9. hier sind manche Produkte schon implementiert und andere geplant [↑](#footnote-ref-9)
10. FR steht für functional requirement [↑](#footnote-ref-10)
11. *NFR* steht für nonfunctional requirements [↑](#footnote-ref-11)
12. Das spezielle Werkzeug ist verantwortlich für die Verwaltung aller diagramlosen Werkzeuge [↑](#footnote-ref-12)
13. Steht für Representational State Transfer [↑](#footnote-ref-13)
14. Steht für Simple Object Access Protocol: Protokoll zum Austausch strukturierter Informationen [↑](#footnote-ref-14)
15. Uniform Resource Identifier [↑](#footnote-ref-15)
16. Single Page Application [↑](#footnote-ref-16)
17. ist die Bezeichnung für ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung. [↑](#footnote-ref-17)